Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Nik Šalej

Razvoj navigacijske naprave za jadralna letala

diplomsko delo

univerzitetni študijski program
računalništvo in informatika

Mentor: prof. dr. Franc Solina

Ljubljana, 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Preučite funkcionalne potrebe za sodobno navigacijsko napravo za jadralna letala s posebnim poudarkom na primernem uporabniškem vmesniku ter izberite primerno strojno in programsko opremo za tako napravo.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
GUI	Graphical user interface	Grafični uporabniški vmesnik
RTOS	Real time operating system	Realnočasovni operacijski sistem
FAI	The World Air Sports Federation	Svetovna letalska zveza
IGC	International Gliding Commission	Mednarodna jadralska komisija
NASA	National Aeronautics and Space Administration	Nacionalna zrakoplovna in vesoljska uprava
SRTM	Shuttle Radar Topography Mission	Topografski projekt
GNSS	Global navigation satellite system	Globalni navigacijski satelitski sistem
GPS	Global positioning system	Globalni sistem pozicioniranja
NMEA	National Marine Electronics Association	Komunacijski standard ameriške mornarice
UTC	Coordinated universal time	Univerzalni koordinirani čas
UART	Universal asynchronous receiver/transmitter	Univerzalni nesinhronizirano sprejemanje in oddajanje

SPI	Serial Peripheral Interface	Sinhrona serijska podatkovna povezava
ADC	Analog-to-digital converter	Analogno digitalni pretvornik
DPI	Dots per inch	Točk na palec
ITO	Indium Tin Oxide	Inidijum tin oksid
LCD	Liquid Cristal Display	Zaslon s tekočimi kristali
PCT	Projected capacitive touch	Projeciran kapacitivni dotik
MMU	Memory management unit	Enota za nadzor pomnilnika
PDA	Personal digital assistant	Prenosni digitalni asistent
FG	Final glide	Dolet
IAS	Indicated air speed	Indicirana zračna hitrost
TAS	True air speed	Dejanska zračna hitrost
GS	Gound speed	Hitrost glede na zemljo
EASA	European Aviation Safety Agency	Evropska agencija za varnost v letalstvu
FAA	Federal Aviation Administration	Zvezna letalska administracija

Povzetek

Naslov: Razvoj navigacijske naprave za jadralna letala

Avtor: Nik Šalej

Diplomsko delo opisuje postopek izgradnje moderne jadralne navigacijske naprave, ki za vnos podatkov uporablja zaslon občutljiv na dotik. Pregledali smo kakšna tehnologija nam je pri tem na voljo, ter kateri tip zaslona je najprimernejši za gradnjo naše navigacijske naprave. Preden smo se loti gradnje, smo si ogledali katere parametre mora naša naprava zajemati in kako jih mora prikazati, da bodo pilotu v pomoč. Nato smo pregledali ključne komponente strojne opreme, ki jo potrebujemo za gradnjo jadralne navigacijske naprave. Na izbrani strojni opremi smo nato zgradili celoten sistem. Ogledali smo si kateri operacijski sistem je najprimernejši, ter kako so implementirane najpomembnejše funkcije naprave. Naprava s še tako dobro strojno opremo je brez učinkovitega grafičnega uporabniškega vmesnika neuporabna. Zato smo se v zadnjem delu ukvarjali, kako najbolj učinkovito predstaviti izmerjene in izračunane podatke na zaslonu občutljivem na dotik. Pregledali smo kakšni so grafični uporabniški vmesniki na napravah namenjenih letalstvu in tistih, ki jih uporabljamo na mobilnih napravah v vsakdanjem življenju in uporabljajo zaslone občutljive na dotik. Dobljeno znanje smo uporabili za gradnjo našega grafičnega uporabiškega vmesnika.

Ključne besede: navigacija, jadralno letalstvo, GUI, GPS, merjenje višine.

Abstract

Title: Development of a navigational device for gliders

Author: Nik Šalej

In the thesis we researched how to develop a modern navigational instrument for gliders with touch screen interface. We have studied what kind of technology and touch screens are available to build such a device. First we checked what kind of data is necessary for a pilot to perform save and pleasant flight. We researched how this data should be displayed. We checked what are critical hardware components. On selected hardware we have build a whole system. We analysed which operating system is suitable and how to implement most important software features for navigational instrument. But even device with the best hardware is not worth anything if we don't have a good graphical user interface. In the last part we examined how we can effectively display measured and calculated values on touch screen. We reviewed graphical user interfaces used on modern avionics and every day mobile devices with touch screens. We combined this knowledge to create our own graphical user interface.

Keywords: navigation, gliding, GUI, GPS, altitude measurement.

Poglavje 1

Uvod

Jadralno letenje je najboljši približek leta ptic. Težavo pa predstavlja človeško telo, saj ne vsebuje čutil, ki bi mu omogočala zaznavanje parametrov, ki so ključni za varno in učinkovito letenje. Prav tako človeško telo ne vsebuje čutil, ki bi omogočila kompleksno zaznavanje tridimenzionalne pozicije v prostoru. Zato je nujna uporaba inštrumentov. V nadaljevanju bomo predstavili, kako zgraditi moderno jadralno navigacijsko napravo, ki za interakcijo z uporabnikom uporablja zaslon občutljiv na dotik. Pri tem se bomo osredotočili na to, kateri so tisti podatki, ki so najpomembnejši za varno, v športnem pogledu pa tudi učinkovito letenje ter kako te podatke prikazati. Pri tem si bomo za vzor vzeli grafične uporabniške vmesnike, prisotne na modernih navigacijskih napravah, ki so nameščene v letalih splošnega letalstva in jadralnih letalih. Ker pa ti grafični uporabniški vmesniki niso prilagojeni za upravljanje z zaslonom občutljivim na dotik, smo se lotili analize grafičnih uporabniških vmesnikov, ki jih uporabljajo moderni pametni telefoni. Dobljeno znanje smo porabili za gradnjo grafičnega uporabniškega vmesnika. Pri tem smo upoštevali kakšni so pogoji, kjer se bo uporabljala ta naprava.

Naprave pa ni brez strojne opreme. Pregledali smo najpomembnejše elemente strojne opreme in predstavili, kako jih bomo vključili v napravo. Pri tem smo se osredotočili predvsem na to kako s posamičnimi komponentami komuniciramo oziroma kako delujejo, da lahko kasneje uporabimo izmerjene

podatke. Vse skupaj smo zaokrožili z implementacijo programske opreme, prilagojene izbranim komponentam. Najprej se je bilo potrebno odločiti, kakšen operacijski sistem bomo uporabili oziroma, če ga sploh bomo uporabili. Ko smo osnovni sistem postavili, smo začeli z vključevanjem najpomembnejših funkcij, ki jih mora imeti moderna jadralna navigacijska naprava. Na koncu smo vse skupaj povezali z že prej omenjenim grafičnim uporabniškim vmesnikom.

Poglavje 2

Jadralno letenje

Jadralno letenje je letenje z jadralnim letalom. Jadralno letalo je zračno plovilo, ki je težje od zraka in za svoje potovanje ne uporablja motorja. Nekatera moderna jadralna letala so sicer opremljena z motorjem, ki pa se uporablja zgolj za poletanje oziroma reševanje iz situacij, ki bi prisilile pilota, da pristane. Jadralna letala so lahko eno ali dvo sedežna.

Obstajata dve vrsti jadralnih letal:

- tista, ki se lahko samo spuščajo. (ang. gliding)
- tista, ki se lahko s pomočjo termike, pobočnih vzgornikov ali zaveternih valov tudi dvigajo. (ang. soaring)

V diplomskem delu bomo opisovali gradnjo moderne navigacijske naprave namenjene tistim letalom, ki se lahko tudi dvigajo. Tem letalom pravimo tudi športna jadralna letala.

2.1 Zgodovina jadralnega letalstva

Jadralno letalstvo kot šport se je začelo razvijati v začetku prejšnjega stoletja. Še posebno pa po prvi svetovni vojni. Zibelka jadralnega letastva je Nemčija.

Razlog za nenadno povečanje zanimanja za jadralno letalstvo je Versajska mirovna pogodba, ki je formalno končala prvo svetovno vojno. V pogodbi so Nemčiji omejili proizvodnjo in uporabo motornih letal, zato so vso znanje vložili v razvoj jadralnih letal [20]. V Nemčiji so prvo tekmovanje organizirali že leta 1920. Na poletnih olimpijskih igrah leta 1936, je bilo jadralno letalstvo demonstracijski šport in so ga po igrah uvrstili kot uradni šport za poletne olimpijske igre leta 1940. V ta namen so skonstruirali posebno letalo, a se šport zaradi začetka druge svetovne vojne na igrah ni pojavil. Zaradi različnih interesov pa se šport ni uvrstil na kasnejše olimpijske igre.

Kljub vsemu pa pod okriljem mednarodne letalske zveze (FAI) potekajo poleg manjših in državnih tekmovanj tudi svetovna in evropska prvenstva. V jadralnem letalstvu ni razlik med spoloma. Tekmovanja ponavadi trajajo od enega do dveh tednov. Zmagovalec je tekmovalec, ki zbere največ točk preko vseh tekmovalnih dni. Vsak tekmovalni dan vodstvo tekmovanja pripravi tekmovalno nalogo, ki je sestavljena iz štartne in ciljne točke ter poljubnega števila obratnih točk med njima. Dnevni zmagovalec je tisti, ki uspe zastavljeno nalogo končati v najkrajšem možnem času. Pri tem mora paziti, da leti zgolj v zračnem prostoru, ki ga je predvidelo vodstvo tekmovanja. Dnevne naloge so dolge od 100 kilometrov pa vse do 1000 kilometrov. Pilot dokazuje obleteno nalogo z zapisovalniki letov GNSS (ang. global navigation satellite system), ki beležijo pozicijo in višino letala med letom. V preteklosti so to počeli s fotografiranjem obratnih točk in barografom. Moderne navigacijske naprave olajšajo navigiranje po zastavljeni nalogi in omogočajo, da se pilot posveti drugim, za letenje bolj pomembnim nalogam. Posledično so se povprečne hitrosti zmagovalcev v zadnjem desetletju precej zvišale.

Poleg tekmovalnega letenja pa piloti letijo tudi prelete. Cilj takšnega letenja je, da v danem dnevu pilot preleti čim večjo razdaljo. Najdaljši preleti so dolgi tudi 3000 kilometrov. Takšne prelete je omogočila sodobna navigacijska oprema, ki pilotu nudi vse potrebne informacije in nadomešča staro, vendar zanesljivo navigacijo s papirnato karto. Tudi pri teh letih piloti svoje dosežke beležijo z zapisovalniki letov GNSS, ki jim omogočajo, da analizi-

rajo svoj let oziroma objavljajo svoje dosežke. Dosežki v jadralnem letalstvu se nagrajujejo z značkami. Pogoje za pridobitev določene značke pa določa mednarodna letalska zveza FAI. Osnovno značko imenovano „Srebrni C“ dobi pilot, ki uspe odleteti vsaj 5 ur, ter ob enem doseči termični prirastek višine vsaj 1000 metrov ter odleteti ravni let dolžine vsaj 50 kilometrov. Določene značke so tudi pogoj za udeležbo na tekmovanjih.

Obstaja nekaj različnih načinov, kako jadralno letalo spraviti v zrak, kjer lahko nato avtonomno nadaljuje z letom. Danes najbolj pogosti metodi sta aerozaprega in vzlet s pomočjo vitle. Vzlet s pomočjo elastike oziroma iz pobočja je danes že skoraj popolnoma opuščena metoda, saj ne zagotavlja dovolj velike začetne energije, bodisi kinetične, bodisi potencialne za uspešno nadaljevanje leta. Nekatera moderna jadralna letala so opremljena z motorjem, ki jim omogoča samostojni polet. Ko jadralno letalo doseže zadostno višino za avtonomno letenje, se motor skupaj s propelerjem pospravi v trup letala, kar omogoča, da takšno letalo leti kot klasično jadralno letalo. Pri vzletu s pomočjo aerozaprege sta vlečno (motorno) in jadralno letalo povezani med seboj z vlečno vrvjo dolžine 50 metrov. Ko jadralno letalo doseže zadostno višino se odpne od vrvi in nadaljuje let s pomočjo vzgornikov. [13]

Pri vzletu s pomočjo vitla vitel postavimo od 1000 do 1500 metrov od jadralnega letala in ju povežemo z jekleno oziroma sintetično pletenico. Vzlet in vzpenjanje sta izredno hitra. Pri tem načinu vzleta se jadralno letalo samodejno odpne od vitla, ko doseže maksimalno višino, ki je v korelaciji z dolžino pletenice.

2.1.1 Jadralno letenje v Sloveniji

Slovenci smo bili od nekdaj v središču razvoja jadralnega letalstva. Še predno sta brata Rusjan izvedla let z motornim letalom leta 1909, je Otmar Kanet že letel z lastnim jadralnim letalom za drsni let. Zgradil je vsaj tri letala. Tudi znameniti Stanko Bloudek je leta 1909 med študijem v Pragi zgradil jadralno letalo. Kmalu za tem je bila izdana tudi prva knjiga v slovenskem jeziku namenjena letenju, kjer so omenjeni tudi prvi poskusi letenja z jadralnimi

letali [37].

V obdobju med obema svetovnima vojnama so v Sloveniji zrastli številni aeroklubi, kjer so deloma po nemških načrtih, deloma pa z lastnimi konstrukcijami začeli z gradnjo jadralnih letal. Po vzoru jadralske šole na Wasserkuppe v Nemčiji, so tudi v Sloveniji začeli organizirano leteli s pobočji, kjer so bile ugodne razmere za doseganje čim daljših letov. Eno najbolj znanih takih središč se je razvilo na Blokah.

Pravi razcvet se je zgodil po drugi svetovni vojni. Na podlagi razpisa iz leta 1946 se je začel razvoj lastnih konstrukcij. Letala so bila po kakovosti najboljša na svetu. Izdelava letal je potekala v tovarni Letov in v Konstrukcijskem biroju letalske zveze Slovenije, ki se je leta 1956 združil v LIBIS. V 70. letih je proizvodnja letal v Sloveniji zamrla. Leta 1978 se je ponovno začela izdelava jadralnih letal iz okrepljene plastične mase v tovarni Elan [13]. V tistem obdobju se je v Slovenij začel tudi razvoj prvih elektronskih pripomočkov za jadralna letala. Kasneje se je v Sloveniji razvoj usmeril na izdelavo dodatne opreme za jadralna letala, sama proizvodnja jadralnih letal pa se je ustavila.

Trenutno je v Sloveniji več kot 10 podjetij, ki se ukvarjajo z jadralnim letalstvom. Na področjih kjer delujejo so v samem svetovnem vrhu in v večini primerov orjejo ledino. Podjetja se ukvarjajo s konstruiranjem in gradnjo jadralnih letal in polproizvodov, električnih pogonskih agregatov, varionavigacijskih naprav in raznovrstne notranje in zunanje opreme za jadralna letala.

2.2 Ključni parametri za varno in učinkovito letenje

Ljudje so v začetku leteli z jadralnimi letali brez inštrumentov. V prvih letih jadralnega letalstva so ljudje leteli tako, da so letalo s pomočjo elastike izstrelili iz vrha hriba. Pilot je nato letalo nekako po občutku poskušal čim dlje zadržati v zraku, a je na koncu vseeno pristal ob vznožju hriba. Šlo je za

relativno kratke lete, kjer so prvenstveno odkrivali aerodinamične zakonitosti letenja.

Kasneje so se z izboljšavami konstrukcij in oblik jadralnih letal, leti bistveno podaljšali in posledično so začeli razvijati tudi inštrumente, ki bi omogočali še daljše, višje in na splošno varnejše lete. Razvoj se je zato usmeril v merjenje hitrosti letala skozi zrak in višine letala med letom. Nastala sta višinomer in merilnik zračne hitrosti.

Z ugotovitvijo, da lahko jadralna letala izkoristijo tudi dvigajoči se zrak v okolici, se je začel razvoj za jadralno letenje najpomembnejšega inštrumenta. To je variometer. Gre za inštrument, ki kaže skoraj trenutno dviganje oziroma spuščanje letala. Poimenujemo ga lahko tudi merilec vertikalne hitrosti. Tak inštrument omogoča, da pilot vstraja dalj časa v zračni masi, ki se dviga in poskuša čim hitreje zapustiti zračno maso, ki se spušča.

Ko letenje v okolici letališča ni bilo zadostno, so v jadralna letala začeli vgrajevati kompase. Pilot se lahko s pomočjo zemljevida, kompasa, hitrosti letenja in štoparice orientira na neznanem terenu. Jadralno letenje je dobilo nove dimenzije. Tehnične omejitve za dolge prelete so s tem odpravili.

Če povzamemo so osnovni parametri, ki omogočajo varno in učinkovito letenje merjenje hitrosti, višine, vertikalne hitrosti, ter orientacija v prostoru. Inštrumenti, ki omogočajo merjenje in prikaz teh parametrov so tudi del minimalne opreme vsakega jadralnega letala. Minimalno opremo jadralnega letala sicer določa Evropska agencija za varnost v letalstvu (EASA) in je predpisana v priročniku vsakega jadralnega letala.

Ko so piloti začeli tekmovati oziroma dokazovati svoje dosežke, se je razvoj usmeril v merjenje in beleženje parametrov leta. Sprva je obstajal samo barograf. Gre za napravo, ki beleži trenutno višino jadralnega letala tako, da igla namočena v črnilo zapiše višino na papir. Po končanem letu lahko pilot tako dokaže višino, ki jo je uspel dosečim med letenjem.

Vstop tehnologije GPS v civilno rabo je omogočil začetek gradnje inštrumentov, ki so poleg zgoraj omenjenih parametrov lahko zaznavali še pozicijo letala v danem trenutku in njegovo hitrost glede na tla. Hitrost glede na tla

(GS) je omogočila merjenje vrste novih parametrov, med katerimi je eden pomembnejših merjenje smeri in jakosti vetra. Podatek o vetru je eden ključnih za izračun doleta. GPS je obenem pilota razbremenil, saj je nadomestil zgoraj opisano navigacijo s pomočjo kompasa. Omogočil je zapisovanje pozicije letala in s tem omogočil, da lahko pilot po končanem letu dokaže, kje je letel.

Pri gradnji našega inštrumenta bomo seveda upoštevali, kateri so najpomembnejši parametri za varno in učinkovito letenje. Te parametre bomo izmerili in jih prikazali na enem mestu na nov in obenem učinkovit način.

2.3 Zgodovina digitalnih inštrumentov za jadralno letalstvo

Predno se posvetimo gradnji inštrumenta, si na kratko oglejmo zgodovino razvoja elektronskih pomagal uporabljenih v svetu jadralnega letalstva.

Začetki segajo v konec 60. let 20. stoletja, ko so se začele razvijati prve električne naprave za jadralna letala. Kot je bilo omenjeno v poglavju 2.2, se je razvoj do prihoda GPS v civilno rabo ukvarjal z izdelavo prvih električnih variometriov. Torej naprav, ki prikazujejo vertikalno gibanje letala. Senzorji v teh napravah so bili na osnovi uporov NTC. Še vedno je šlo za analogno tehniko. So pa bili taki variometri že opremljeni z zvočnikom, ki je pilota s piski obveščal, kakšna je trenutna vertikalna hitrost letala. S tem je pilot lahko lažje iskal dviganja in več časa opazoval okolico, ter le bežno preveril stanje merilnikov na inštrumenti plošči.

Leta 1985 so se pojavile naprave, ki so jih že poganjali mikrokontrolerji. Takšne naprave so že bile sposobne izvajati določene dodatne izračune. Te so prikazovali na 2x8 segmentnih zaslonih LCD. Senzorji za zajem podatkov so bili še vedno analogni.

Leta 1992 je GPS prešel iz vojaške tudi v civilno uporabo. To možnost so takoj izkoristili proizvajalci elektronskih naprav za jadralna letala, ki so do tedaj v večini izdelovali zgolj elektronske variometre. Prihod GPS je pomenil, da so naprave postale tudi navigacijske. Naenkrat je bilo možno

izračunati ogromno novih podatkov, ki jih je bilo potrebno prikazati uporabniku. Segmentni zaslon so tako nadomestili monokromatski grafični ekrani z resolucijo 128x64 točk.

Leta 1996 se je zgodil tudi preboj na senzorjih. Analogne senzorje za hitrost in višino so nadomestili digitalni, kar je precej izboljšalo resolucijo meritev, predvsem pa olajšalo proizvodnji proces.

Leta 2007 pa so začeli nastajati prvi sistemi na osnovi linux operacijskega sistema z velikimi barvnimi zasloni. Takšne naprave so tudi trenuto na tržišču.

Poglavje 3

Strojna oprema

Pri razvoju moderne navigacijske naprave je ključna izbira ustrezne strojne opreme. Posamične komponente strojne opreme nato s pomočjo programske opreme dopolnimo tako, da naprava predstavlja zaključeno celoto. V nekaj naslednjih poglavjih si bomo ogledali, kateri so ključni elementi strojne opreme. Pri tem se bomo osredotočili predvsem na to, za kaj se uporabljajo ter kako bomo v naši programski opremi zagotovili ustrezno komunikacijo oziroma implementacijo. Ne bomo pa podrobneje opisovali, kako so elementi med seboj povezani na samem tiskanem vezju.

3.1 Senzorji

3.1.1 GNSS

GNSS je kratica za globalni navigacijski satelitski sistem. Trenutno samo dva sistema zagotavljata globalno pokritost s signalom. To sta ameriški NAVSTAR GPS in ruski GLONASS. Svoje sisteme razvijajo tudi Kitajska (BeiDou), Indija (NAVIC) in Evropska unija (Galileo). Ti sistemi so zaenkrat le delno operativni oziroma imajo zgolj lokalno pokritost s signalom. [11]

V naši napravi smo se zato odločili podpreti ameriški GPS. Na trgu je kar nekaj modulov, ki podpirajo sprejem signala GPS. Razlikujejo se predvsem v številu kanalov, ti pa definirajo koliko satelitov hkrati lahko sprejemnik GPS

uporabi za izračun pozicije. Večje kot je število satelitov, ki jih lahko uporabijo, boljši izračun pozicije bo izvedel sprejemnik GPS. Vsem sprejemnikom GPS je skupna komunikacija preko protokola NMEA.

GPS je bil v začetku razvit za mornarico in zato za komunikacijski protokol med sprejemnikom in aplikacijo skrbi ameriška državna pomorska organizacija. Obstaja več standardov protokola NMEA. Kasnejši standardi so bili razviti predvsem za komunikacijo med različnimi elektronskimi napravami na ladjah. Za komunikacijo s sprejemnikom GPS pa zadostuje NMEA0183.

Sama komunikacija s sprejemnikom GPS poteka preko vmesnika UART. Podatki se nato prenašajo s protokolom NMEA. Protokol je podrobneje predstavljen v naslednjem podpoglavju.

3.1.2 Protokol NMEA

Protokol NMEA je sestavljen iz tako imenovanih stavkov NMEA. Dovoljeni so samo znaki iz kodne tabele ASCII. Poleg standardnih stavkov NMEA proizvajalci pogosto uporabljajo lastne stavke. Vsak tak stavek se namesto z znakom '$' začne z '$P'. [24]

Tipična sestava stavka NMEA je:

- Vsak stavek se začne z znakom '$'.
- Sledi 5 znakov, ki identificirajo stavek. Od tega prva dva določata napravo, ki pošilja stavek, trije pa tip stavka.
- Sledijo polja z vsebino.
- Vsebina je ločena z vejicami. Če vsebina ne obstaja, izbrano polje ostane prazno, vejica pa je vseeno prisotna.
- Zvedica ('*') pomeni začetek konca stavka. Zvezdici vedno sledi checksum, ki je je sestavljen kot ekskluzivni OR preko vseh kod znakov ASCII, med znakom za začetek stavka ('$') in zvezdico ('*').

- Checksum ni obvezen podatek za vse vrste sporočil. Stavki, ki pa jih pošilja sprejemnik GPS (RMA, RMB in RMC) pa spadajo v skupino tistih, kjer mora biti checksum vedno prisoten. Kadar checksum ni prisoten, se zvezdica ('*') ne pošlje.
- Sporočilo se konča z $< CR >< LF >$.

Primer standardnih stavkov NMEA, ki jih pridobimo iz našega sprejemnika GPS si bomo ogledali spodaj [23]. Poleg prikazanih stavkov NMEA je na voljo kopica drugih stavkov, a jih v naši napravi ne bomo potrebovali. Poleg standardnih stavkov NMEA proizvajalci sprejemnikov GPS uporabljajo še lastne stavke NMEA.

GGA

$GPGGA - Global Positioning System Fix Data
$GPGGA,hhmmss.ss,llll.ll,a,yyyyy.yy,a,x,xx,x.x,x.x,M,x.x,M,x.x,xxxx*hh

Pozicija	Vrednost	Opis
1	123456	Trenutni UTC čas (12:34:56)
2	4607.038	Geografska širina (46° 07.038')
3	N	N (severna polobla) ali S (južna polobla)
4	01431.000	Geografska dolžina (14° 31.000')
5	E	E (vzhodno od meridiana) ali W (zahodno od meridiana)
6	1	Kvaliteta signala GPS (0=neveljaven; 1=Veljaven; 2=Veljaven diferencialni signal)
7	09	Število satelitov v uporabi (ne število vidnih)
8	0.9	Horizontalna napaka signala
9	244.2	Višina antene nad/pod srednjim morskim nivojem (geoid)
10	M	Meter (Enota za podajanje višine)
11	46.7	Razlika višine med Geoidom (srednjim morskim nivojem) in modelom WGS-84 Geoidal separation

12	M	Meter (Enota za podajanje razlike višine)
13	prazno polje	Starost podatka v sekundah od zadnje posodobitve diferencialne referenčne postaje.
14	prazno polje	Identifikacijska številka difrencialne referenčne postaje
15	*56	Checksum

RMB

$GPRMB - Recommended minimum navigation info

$GPRMB,A,x.x,a,c–c,d–d,llll.ll,e,yyyyy.yy,f,g.g,h.h,i.i,j*kk

Pozicija	**Vrednost**	**Opis**
1	A	Status podatkov (A = veljavni; V = neveljavni)
2	0.66	Napaka izračuna smeri v navtičnih miljah (največ 9.99NM)
3	L	Smer popravka napake (L = levo; R = desno)
4	003	Identifikacijska številka začetne točke
5	004	Identifikacijska številka konče točke
6	4917.24	Geografska širina končne točke
7	N	N (severna polobla) ali S (južna polobla)
8	12309.57	Destination waypoint longitude
9	W	E (vzhodno od meridiana) ali W (zahodno od meridiana)
10	001.3	Razdalja do končne točke v navtičnih miljah
11	052.5	Bearing na končno točko v stopinjah (pravi sever)
12	000.5	Hitrost v smeri končne točke v volzih
13	V	Status prihoda (A = prihod, V = ni prihoda)
14	*20	Checksum

RMC

$GPRMC - Recommended minimum specific GPS/Transit data
$GPRMC,hhmmss.ss,A,llll.ll,a,yyyyy.yy,a,x.x,x.x,ddmmyy,x.x,a*hh

Pozicija	Vrednost	Opis
1	123519	Trenutni UTC čas (12:35:19)
2	A	Status podatkov (A = veljavni; V = neveljavni)
3	4807.038	Geografska širina
4	N	N (severna polobla) ali S (južna polobla)
5	01131.000	Geografska dolžina
6	E	E (vzhodno od meridiana) ali W (zahodno od meridiana)
7	022.4	Hitrost glede na tla (GS) v vozlih
8	084.4	Smer poti glede na tla v stopinjah (pravi sever)
9	230316	Datum (23. marec 2016)
10	003.1	Magnetna deklinacija v stopinjah (Zahodne vrednosti se prištevajo pravi smeri, vzhodne pa odštevajo)
11	W	Smer magnetne deklinacije (E = vzhodno od meridiana, W = zahodno od meridiana
12	*6A	Checksum

3.1.3 Senzor višine

Kot smo že uvodoma povdarili, je podatek o višini eden najbolj ključnih za pilota. Podatek lahko uporabimo za določanje tridimenzionalne pozicija v prostoru, kar je osnova za vse nadaljne izračune. Natančen podatek o višini je zato zelo pomemben.

Poznamo več metod merjenja višine. Razlikujejo se predvsem po tem, kakšen podatek o višini nam dajo. Ene nam podajo podatek o višini glede na referenčno točko, druge pa relativno višino glede na površje Zemlje. V poglavju 3.1.1 smo opisali sprejemnik GPS. Za merjenje višine bi tako lahko uporabili kar podatek iz sprejemnika GPS. Težava te meritve je, da je na-

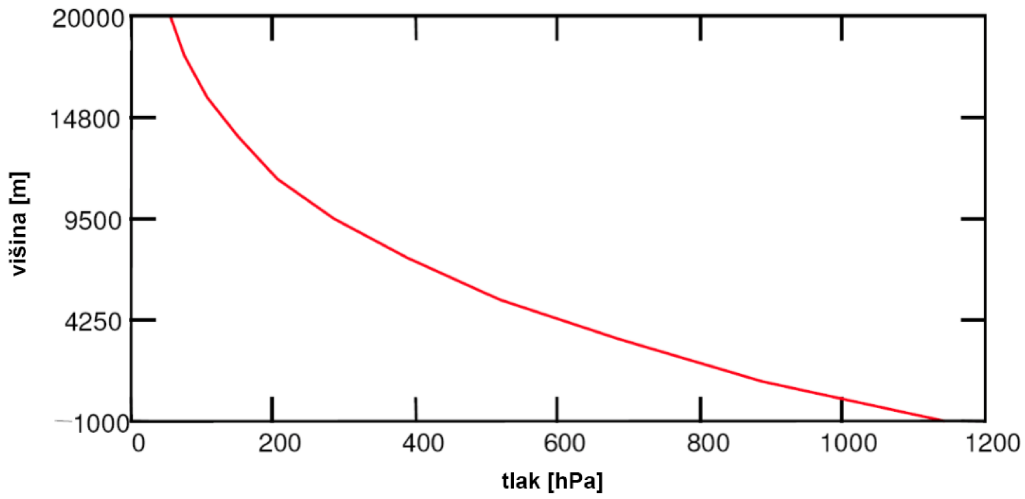
tančnost pogojena s številom sprejetih satelitov. Poleg tega je bil sistem GPS zasnovan za dvo dimenzionalno pozicioniranje. Lega satelitov mu zato ne omogoča dovolj natančnih meritev.

Drug način za merjenje podatkov je radarski sistem. Tak sistem lahko vsebuje laserski ali ultrazvočni oddajnik/sprejemnik, ki meri čas med oddajo signala in sprejemom odbitega signala od Zemlje. Poleg energetske potratnosti, je potrebno tak senzor dobro umeriti in deluje zgolj ob idealni legi letala v zraku ter je popolnoma neprimeren za mobilno navigacijsko napravo. Poleg tega je podatek, ki ga dobimo zgolj relativna višina nad terenom, zato potrebujemo za podatek o absolutni višini tudi višino terena nad mestom meritve.

Najpogostejša metoda za merjenje višine je zato merjenje višine na podlagi spremembe zračnega tlaka. Zračni tlak v okolici se lahko spremeni zaradi spremembe vremena v okolici, spremembe temperature ali/in spremembe višine. Nas seveda zanimajo odčitki, ki so posledica spremembe višine, a moramo poznati tudi druge pogoje, ki lahko vplivajo na meritev. Spremembo vremena obravnavamo kot motnjo za meritev. Ob spremembi vremena moramo za ustrezno meritev tako poznati trenuten referenčni zračni tlak. Podatek o tem dobimo pri vremenski napovedi ali kontroli zračnega prometa. Višino vedno merimo nad neko referenčno zračno ploskvijo. Pogrešek meritve zaradi spremenjene temperature lahko odstranimo, saj naš tlačni senzor za višino vsebuje tudi senzor temperature.

Pri implementaciji višinskega senzorja smo se tako največ ukvarjali s spremembami tlaka, ki so posledica spremenjene višine leta. V višinskem območju, kjer leti jadralno letalo se tlak z višino spreminja nelinearno. Nelinearnost spremebe tlaka z višino je vidna na sliki 3.1. Predstavljena nelinearnost velja ob izhodiščnih pogojih, ki določajo standardno atmosfero ICAO [16]. Pri tem velja, da je zračni tlak na nadmorski višini 0 metrov 1013,25 hPa, temperatura 15°C, ter temperaturni gradient 0.65°C na 100 metrov do višine 11 kilometrov.

V bližini morske gladine zračni tlak pade za 7 Pa na vsakih 100 metrov.



Slika 3.1: Sprememba tlaka z višino

Natančnejši opis spremebe tlaka z višino pa podaja enačba 3.1. Enačba velja zgolj v delu atmosfere, ki mu rečemo troposfera. Ta sega do višine 11 000 metrov. To območje pa povsem zadošča potrebam naše naprave.

$$h = \frac{T_0}{T_{gradient}} * \left\{ 1 - \left(\frac{p}{P_0} \right)^{T_{gradient} * \frac{R}{q}} \right\} \tag{3.1}$$

kjer je:

h - izračunana višina v metrih

T_0 - temperatura na višini $0m$ (288,15 K)

$T_{gradient}$ - temperaturni gradient ($0{,}65K/100m$

P_0 - tlak na višini $0m$ (1013,25 hPa)

p - trenutni tlak

R- specifična konstanta plina ($287{,}052[J/(K * kg)]$) g - gravitacijski pospešek (9,806 m/s^2)

Digitalni senzorji za tlak se razlikujejo po merilnem območju in načinu

merjenja. Glede na način merjenja poznamo senzorje, ki merijo diferenčni tlak in takšne, ki merijo absolutnega. Sami smo izbrali takšnega, ki meri absolutnega. Senzor, ki smo ga izbrali je prilagojen za delovanje, ko je tlak v okolici med 0 in 1013 hPa. Senzor je tovarniško kalibriran. Vsebuje piezouporovni senzor tlaka in analogno-digitalni pretvornik. Na izhodu lahko preberemo 16-bitne vrednosti tlaka in temperature. Za komunikacijo uporabljamo 3-žično serijsko komunikacijo. Senzor višine nam podaja izredno dobre relative meritve. Absolutna napaka meritve pa je manj kot 5 metrov. [36]

V meritvah se pojavi tudi šum. Šum bi lahko odstranili s povprečenjem večjega števila meritev. Takšna odstranitev šuma je dobra, ko želimo podatek o višini v nekem daljšem časovnem obdobju. V jadranem letalu pa želimo natančen podatek o višini v vsakem trenutku. To smo rešili z uporabo filtra [36], ki ga opisuje enačba 3.2. Filter je nelinearen in upošteva N predhodnih meritev. S številom predhodnih meritev, ki jih upoštevamo določamo tudi odzivnost filtra. Več meritev kot upoštevamo počasneje bomo dobili odčitek spremembe, ki je dejanska posledica sprememe višine oziroma zračnega tlaka. V praksi se je takšen filter izkazal kot zelo dober.

$$h = \frac{h_{nova} + h * KOEFICIENT}{KOEFICIENT + 1} \tag{3.2}$$

h - izračunana višina v metrih

h_{nova} - nova meritev višine

Senzor višine pa ni uporaben samo za ugotavljanje trenutne višine letala v zraku, ampak omogoča tudi ugotavljanje spremembe te višine v določenem času. Takšni napravi pravimo variometer. Uvodoma smo povdarili, da je variometer najpomembnejši inštrument vsakega jadralnega pilota, saj mu poenostavi letenje. Opraviti moramo dovolj meritev, ki jih s pomočjo ustreznega filtra zgladimo, da odstranimo šum. V naši napravi izvajamo do 50 meritev na sekundo. Vse to nam omogoča, da s pomočjo podatka o trenutni višini pridelamo tudi podatek o vertikalni hitrosti jadralnega letala.

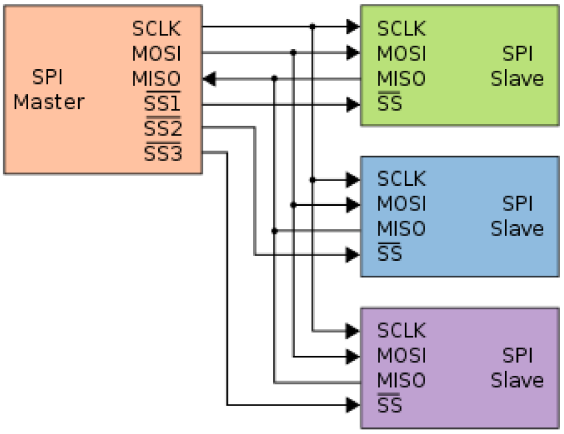
3.2 Hramba podatkov

Vsaka jadralna navigacijska naprava potrebuje za svoje delovanje tudi spomin, ki ohrani podatke tudi po izklopu naprave. V ta spomin se shranjujejo leti in uporabniške nastavitve. Zaradi poenostavitve implementacije in nizke cene, smo se v našem primeru odločili uporabiti kartico SD kot medij za shranjevanje podatkov.

Kartica SD je spominska kartica, ki so jo za potrebe prenosnih elektronskih naprav razvila podjetja Panasonic, SanDisk in Toshiba, sedaj pa za njen razvoj skrbi neprofitna organizacija SD Association [32] . Ta skrbi za nove standarde, ki so posledica povečanih kapacitet in hitrosti prenosa novejših SD kartic. Kartica SD se je sicer razvila iz MMC. Večina naprav, ki podpirajo kartice SD so tako kompatibilne tudi z MMC. Kartica MMC je bila prva spominska kartica, ki je za komunikacijo uporabljala serijsko komunikacijo. Posledično je bila lahko kartica precej manjša, z manjšim povezovalnim vmesnikom.

Kartica SD je sestavljena iz flash pomnilnika in miktrokontrolerja, ki skrbi da se podatki na kartici berejo in pišejo. Obenem mikrokontroler izvaja detekcijo napak in preprečuje prekomerno uporabo določenih pomnilniških celic. Prenos podatkov med kartico SD in nadrejeno napravo poteka v blokih. En blok je vedno velik 512 bajtov, ne glede na to, če želimo dostopati do podatka, ki je manjši. To lastnost s pridom uporabljamo pri prenašanju večjih datotek. V pomnilniku naprave predvidimo prostor za prenos bloka in vedno zahtevamo prenos zgolj 512 bajtov. Kartica SD v našem sistemu je velika 8 GB in zato uporablja datotečni sistem FAT32.

S kartico SD lahko komuniciramo na več načinov. V naši implementaciji smo uporabili komunikacijo preko vodila SPI [33]. Vodilo SPI je standard za sinhrono serijsko povezavo elektronskih naprav in deluje v polnem dvosmernem načinu. Dvosmernost je dosežena s principom nadrejene/podrejene naprave (ang. master/slave). Pri tem lahko isto vodilo istočasno uporablja več naprav. Ena izmed njih je nadrejena, vse ostale so podrejene. Komunikacija lahko poteka samo med nadrejeno in izbrano podrejeno napravo. Za



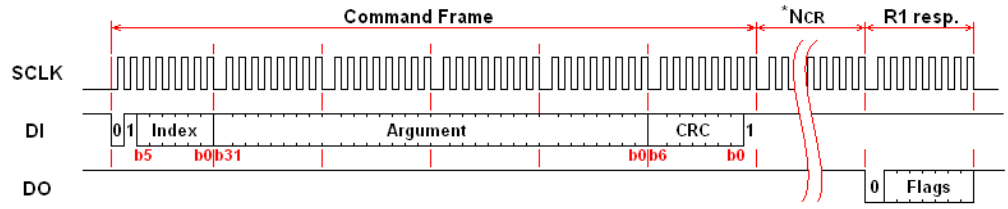
Slika 3.2: Ena nadrejena naprava v povezavi s tremi podrejenimi. Tipičen primer uporabe vodila SPI.

komunikacijo se uporabljajo štiri signalne linije:

- MISO - nadrejena naprava sprejema, podrejena pošilja,
- MOSI - nadrejena naprava pošilja, podrejena sprejema,
- SCK - urin takt (določa ga nadrejena naprava),
- SS - signal za določanje, s katero podrejeno napravo komunicira nadrejena naprava (aktiven v nizkem stanju).

Vodilo SPI zahteva štiri povezave. To skupaj s preprosto zasnovo vodila omogoča izredno hitre prenose podatkov. Tudi do 20 Mbps. Na sliki 3.2 je shema povezave SPI, kjer si eno vodilo delijo tri podrejene naprave ena nadrejena pa s signalom SS določa, katera naprava je aktivna.

Ker v privzetem načinu kartica SD ne deluje v načinu SPI, jo je potrebno v to najprej prepričati. To storimo tako, da linji MOSI in SS postavimo na logično stanje 1. V tem stanju moramo počakati vsaj 74 ciklov signala SCK [1].



Slika 3.3: Zgradba ukaza za komunikacijo s kartico SD, preko vodila SPI.

Kasneje postavimo signal SS na 0 in pošljemo ukaz CMD0. Ta ukaz kartico SD pripravi, da začne komunicirati v načinu SPI. Kasneje z kartico SD komuniciramo z ukazi. Vsak ukaz je dolg 48 bitov. Sestavljen je iz dveh začetnih bitov (01). Sledi 6 bitov, ki definirajo ukaz. Naslednjih 32 bitov je na voljo za morebitne dodatne informacije, ki jih potrebujejo določeni ukazi. Sledi 7 bitni CRC, ki pa ga kartica SD v privzetem načinu ignorira. Vse skupaj zaključi en končni bit (1).

3.3 Zaslon občutljiv na dotik

Zaslon občutljiv na dotik je vhodna naprava, položena na zaslon, ki omogoča vnos podatkov s pomočjo enega ali več prstov. Vnos lahko naredimo tudi s posebnimi pisali [34].

Zaslon občutljiv na dotik omogoča uporabniku neposredno manipulacijo z grafičnim uporabnišlim vmesnikom. To seveda definira popolnoma nov način gradnje grafičnih uporabniških vmesnikov. Pri tem je potrebno imeti v mislih predvsem, kakšne so velikosti gradnikov, ter kje so pozicionirani.

Človeški prst je relativno natančna naprava in lahko izvaja gibe do 0.1mm natančno [10]. Težava pa je, da s takšno natančnostjo ne zmore zadeti določenega grafičnega elementa prikazanega na zaslonu. Težava pravzaprav ni v prstu samem ampak v očeh oziroma sami povezavi med očmi, možgani in prsti. Pri očeh pride to tako imenovane paralakse, ki je možgani ne morejo povsem odstraniti. Paralaksa je efekt, ko vsako oko vidi prst pod rahlo

drugačnim kotom. Natančnost pa se spreminja tudi glede na pozicijo pritiska na zaslonu samem. Največja je natančnost v srednejm delu zaslona, najslabša na robovih.

Poznamo več vrst zaslonov občutljivih na dotik. V naslednjih poglavjih si bomo ogledali dva najbolj pogosta predstavnika dveh osnovnih skupin zaslonov občutljivih na dotik. Na podlagi tega se bomo odločili kateri je primernejši za našo implementacijo. Pri tem s pojmom zaslon občutljiv na dotik mislimo predvsem na tisti del zaslona, ki deluje kot vhodna naprava, sam zaslon, torej izhodno napravo, pa pri tem zavestno ignoriramo.

3.3.1 Rezistivni zasloni

Rezistivni zaslon občutljiv na dotik je zgrajen iz dveh upogljivih plasti, ki sta prekriti s polprevodnim materialom. Plasti sta med seboj rahlo oddaljeni. Plasti sta sestavljeni iz stekla oziroma kakšne druge transparentne plastične mase in sta na notranji strani premazani z ITO (ang. Indium Tin Oxyde). Razvoj takšnih zaslonov se je začel že v 60-ih letih 20. stoletja. Obstaja več vrst rezistivnih zaslonov, ki pa se v svoji osnovni zgradbi ne razlikujejo med sabo. Najbolj pogosta sta 4 in 5 žični rezistivni zaslon [7].

Osnova za merjenje pozicije vsakega rezistivnega zaslona je merjenje upornosti na dveh oseh. To storimo tako, da najprej napajamo os Y zaslona z napajalno napetostjo. Ob pritisku na zaslon lahko na eni od elektrod na X osi izmerimo napetost. Napetost s pomočjo ADC pretvorimo v numerično vrednost. Natančnost meritve je seveda odvisna od ADC. Tipično pa rezistivni zasloni omogočajo zelo visoke resolucije (4096 DPI). Postopek meritve nato ponovimo še za os X, tako da napajalno napetost pripeljemo na os X in merimo vrednosti na osi Y. Krmilnik takšnega zaslona tako zahteva zgolj mehanizem za preklapljanje med branjem in napajanjem obeh kooridinat ter pretvarjanjem meritev s pomočjo ADC.

Surove meritve pozicije pritiska na zaslonu so podvržene zunanjim dejavnikom. Da bi se jim izognili, so se uveljavile metode povprečenja meritev. Ker so takšne meritve hitre, lahko sistem opravi veliko meritev, ki jih nato

uporabi za izboljšanje rezultatov. Zaporedne meritve, ki jih upravlja krmilnik zaslona se ne smejo razlikovati za več kot v naprej določeno vrednost. S tem se izognemo, da bi spremembo položaja na zaslonu upoštevali kot šum. Če padejo znotraj mej, jih povprečimo, da odstranimo šum. To ponovimo za obe osi, tako dobimo dve neodvisni povprečni vrednosti za obe osi. Iz teh sestavimo pare koordinat in na parih koordinat ponovimo zgoraj opisan postopek. Če kateri koli par koordinat pade iz naprej predvidenih mej, se vse meritve za obe osi izbrišejo, celoten postopek pa se ponovi. Na koncu tako dobimo pozicijo pritiska na zaslonu, ki smo jih skoraj popolnoma odstranili šum.

Je pa potrebno vsak rezistivni zaslon umeriti, ko ga staknemo skupaj z zaslonom, ki prikazuje sliko. Umerjanje poteka tako, da s pisalom ali prstom pritisnemo na označena mesta na zaslonu. Ponavadi to storimo na 5 točkah. Ustvarimo si pare, ki povezujejo meritev pritiska na zaslon in koordinate izpisa točke na zaslonu. To znanje izkoristimo, da surove vrednosti, ki nam jih poda ADC pretvorimo v koordinatni sistem točk na zaslonu.

3.3.2 Kapacitivni zasloni

Elektrode na vsaki strani zaslona na prevodnem sloju držijo natančno količino električnega naboja, ki sestavlja enakomerno električno polje. Ko se uporabnik dotakne takšnega zaslona, se delček naboja prenese na uporabnika, kar lahko zaznamo. To zmanjšanje naboja merimo z namenskim vezjem, ki mertive s pomočjo ADC pretvori v kordinate. Te so ponavadi kar enake točkam na zaslonu, zato takšnega zaslona občutljivega na dotik ni potrebno kalibrirati skupaj z zaslonom.

Najpogostejša implementacija kapacitivnega zaslona občutljivega na dotik je tako imenovana projecirana kapacitivnost (PCT) [6]. Zasloni so zgrajeni iz dveh serij vzporednih tankih elektrod, ki so med seboj pravokotne ali pa iz dveh plasti prevodnega materiala, ki skupaj tvorita pravokotno matriko. Elektrode tvorijo mrežo kondenzatorjev. Velika gostota elektrod omogoča zazanavanje dotikov brez fizičnega dotika takšnega zaslona. To nam omogoča,

da takšen zaslon prekrijemo z zaščitnim steklom, kar varuje zaslon pred praskami, prahom ali vodo in samemu zaslonu poveča življenjsko dobo ter poveča robustnost.

Ko uporabnik pritisne na zaslon s prstom ali kakršnim koli drugim prevodnim materialom, ki ima dielektrično lastnost različno od zraka, se spremeni kapacitivnost kondenzatorjev v bližini dotika. Na podlagi spremembe lahko določimo mesto dotika.

Obstajata dve vrsti zaslonov PCT, ki se razlikujeta zgolj po implementaciji kapacitivnih elektrod. Za večdotično podporo je uporabna zgolj implementacija, kjer vsako presečišče elektrod predstavlja posamičen kondenzator. Napetost pripeljemo na eno izmed osi. Ko se s prstom ali drugim prevodnim materialom približamo zaslonu, lahko merimo spremembo kapacitivnosti na vsaki posamični točki. Tako lahko natančno določimo mesto dotika in omogočimo zaznavanje večih prstov.

Takšne zaslone uporabljajo vsi moderni pametni telefoni občutljivi na dotik. Kapacitivni zasloni imajo manj težav s prosojnostjo, saj dve prevodni plasti nadomestimo z eno samo. Še vedno pa nam ostane zračna plast med zaslonom in steklom, ki je del zaznavanja dotika. Tudi za to obstaja rešitev, kjer oba elementa zlepimo med sabo (ang. Optical bonding). S tem povečamo vidljivost na zaslon, saj odstranimo dve plasti, kjer bi lahko prišlo do odboja svetlobe [25].

3.3.3 Primerjava zaslonov in izbira ustreznega za našo implementacijo

Spodaj so naštete prednosti in slabosti obeh vrst zaslonov občutljivih na dotik [29]. Predno smo se lotili implelemtacije smo izbrali tip zaslona, ki najbolj ustreza zahtevam naše naprave.

Prednosti rezistivnega zaslona občutljivega na dotik:

- Nizka cena.

- Visoka odportnost na vodo in prah.
- Uporaben za dotike s prsti, pisalom in rokavicami.

Slabosti rezistivnega zalosna občutljivega na dotik:

- Ni dovolj občutljiv na dotike. Za zaznavo dotika je potrebno močno pritisniti na zaslon.
- Zmanjšuje kontrast zaradi dveh dodatnih plasti. Med plastmi se svetloba odbija pod različnimi koti.
- Zmanjša svetilnost zaslona.
- Ne omogoča večdotičnosti.

Prednosti kapacitivnega zaslona občutljivega na dotik:

- Visoka občutljivost na dotik. Za natančno izbiro ni potrebno pisalo.
- Omogoča večdotično zaznavanje.
- Omogoča prekrivanje z zaščitnim steklom, folijo ali premazom.
- Večja prosojnost in ostrina slike.

Slabosti kapacitivnega zaslona občutljivega na dotik:

- Za zaznavo dotika je potrebno uporabiti prevoden material, ne deluje z rokavicami.
- Višja cena zaradi kompleksnosti izdelave.
- Zaslon je zaradi ene same plasti stekla bolj lomljiv.

Ker gradimo moderno jadralno navigacijsko napravo, kjer je najpomembnejša lastnost vidnost zaslona v slabših svetlobnih pogojih, smo se odločili za kapacitivni zaslon občutljiv na dotik. Možnost leplenja zaslona in ena sama dodatna steklena površina izboljša vidljivost v primerjavi z rezistivnim zaslonom. Dodaten razlog za izbiro kapacitivnega zaslona občutljivega na dotik je, da tak zaslon uporabljaja večina modernih pametnih telefonov. Tam so večdotične kretnje zelo pogoste. To možnost želimo uporabiti tudi pri gradnji našega grafičnega uporabniškega vmesnika.

Kapacitivni zaslon, ki smo ga izbrali je opremljen z mikrokontrolerjem. Ta skrbi za zaznavanje dotikov. Komunikacija z mikrokontrolerjem poteka preko vodila SPI. Mikrokontroler sporoča dotike kar v koordinatah, ki so ekvivalentne resoluciji zaslona, zato je poenostavljeno preračunavanje dotikov v točke na zaslonu. S tem smo se izognili tudi potrebi po kalibraciji vsakega zaslona.

Izbrani kapacitivni zaslon omogoča zaznavanje 5 dotikov hkrati. V naši aplikaciji bomo uporabili samo eno in dvo dotične uporabniške dotike. Mikrokontroler poleg dotikov zaznava tudi kretnje, kar nam olajša njihovo zaznavanje. Ko se zgodi dotik ali kakšna kretnja, mikrokontroler pošlje prekinitveno zahtevo, s katero sporoča, da je na voljo nov podatek. Podatek nato preberemo preko vodila SPI in ga uporabimo za manipulacijo grafičnega uporabniškega vmesnika.

Poglavje 4

Programska oprema

V tem poglavju si bomo ogledali, kako smo implementirali programsko opremo na prej opisani strojni opremi. Pri tem se bomo osredotočili na ključne elemente razvoja programske opreme za našo napravo. Najprej bomo pregledali, kakšen operacijski sistem smo izbrali, nato pa kako smo implementirali najpomembnejše funkcije jadralne navigacijske naprave.

4.1 Operacijski sistem

Prvi problem na katerega smo naleteli pri implementaciji programske opreme je izbira operacijskega sistema. Le ta določa ves nadalnji razvoj. Izbira ustreznega operacijskega sistema je ključna, saj bi kasnejša sprememba povzročila ogromno dodatnega dela.

Izbrati smo morali operacijski sistem, ki bo deloval na 32 bitnem mikroprocesorju iz družine ARM Cortex M4, ki teče pri 180 MHz in bo imel na voljo do 32 MB SDRAM zunanjega pomnilnika. Izbran mikroprocesor je brez strojne podpore za nadzor pomnilnika MMU (ang. Memory Management Unit), kar precej zmanjša različne možnosti, ki jih imamo pri izbiri operacijskega sistema [21]. MMU namreč skrbi, da opravila ki tečejo na sistemu dobijo dostop do dela pomnilnika, ki ga potrebujejo za svoje delovanje. Enota skrbi za pretvorbo virtualnih pomnilniških naslovov v fizične, za

predpomenje in varovanje alociranih delov pomnilnika pred neavtoriziranim dostopom. Enoto MMU bi sicer lahko implementirali kot zunanjo enoto ali pa programsko, a sta obe rešitvi v našem primeru nesprejemljivi. Prva bi povečala kompleksnost sistema in dodala nepotrebne stroške, druga pa bi porabljala dragocen procesorski čas, ki ga potrebujemo za druga opravila.

V naslednjih poglavjih si bomo ogledali možnosti, ki jih imamo pri izbiri operacijskega sistema za našo napravo. Na koncu bomo predstavili za katero možnost smo se odločili in zakaj.

4.1.1 Klasični operacijski sistemi

Naš pregled bomo začeli z klasičnimi operacijskimi sistemi [26]. S pojmom klasičen operacijski sistem bomo opisovali operacijske sisteme, ki jih uporabljamo v vsakdanjem življenju. Gre za operacijske sisteme, ki tečejo na modernih pametnih telefonih, tabličnih in namiznih računalnikih, PDA in navigacijskih napravah. Na teh napravah sicer tečejo precej različne vrste različnih operacijskih sistemov, a so si vseeno dovolj podobni, da jih lahko opisujemo skupaj. Vsem je enotno, da delujejo kot vmesnik med strojno opremo in uporabniškimi aplikacijami. Ostale skupne lastnosti so, da opravilom, ki tečejo:

- dodeljujejo in varujejo pomnilniški prostor,
- omogočajo dostop do vhodno/izhodnih naprav,
- dodeljujejo procesorski čas,
- jih časovno razporejajo.

Poleg tega pa izvajajo reševanje konfliktov pri dostopu do virov ter optimizirajo in nadzirajo dostop do njih. Uporabnikom omogočajo interakcijo s sistemom. Vsem je skupno, da so večopravilni in za interakcijo z uporabnikom uporabljajo grafični uporabniški vmesnik (GUI).

Ti operacijski sistemi so torej precej kompleksni, a obenem poenostavijo razvoj uporabniškega dela aplikacije. Tam nam zato ni potrebno skrbeti za

vse podrobnosti v zvezi s strojno opremo. Ti operacijski sistemi so razviti za precej zmogljivejšo strojno opremo kot je naša. Poleg tega v njih najdemo ogromno orodij, ki jih v našem primeru ne potrebujemo, a jih iz samega sistema ne moremo odstraniti. Največjo težavo pa predstavlja že zgoraj omenjeno pomanjkanje enote MMU. Dodeljevanje, varovanje in predpomnenje pomnilnika je ena ključnih lastnosti operacijskega sistema.

V iskanju klasičnega operacijskega sistema, ki omogoča vse prej naštete lastnosti, a pri tem ne potrebuje enote MMU smo rešitev našli v uClinuxu [18]. Obstaja kar nekaj implementacij uClinux operacijskega sistema na podobno strojno opremo, kot jo imamo sami. Glavna razlika med uClinux in klasičnim Linux operacijskim sistemom je, da lahko deluje brez zgoraj omenjene enote MMU. To pa pomeni, da ne omogoča uporabe navideznega pomnilnika, kar odvzame precejšen del abstrakcije pomnilnika, ki jo uporabljajo operacijski sistemi. Zato mora vsako opravilo v uClinuxu zasesti svoje mesto v pomnilniku, kar pomeni, da se vse spremembe dejansko odražajo neposredno v fizičnem pomnilniku.

To pa predstavlja precejšne posledice pri razvoju programske opreme. Za progam, ki ga bomo napisali in poganjali na takšnem sistemu moramo že v času prevajanja poznati njegove potrebe po pomnilniku. To pomeni, da ne smemo uporabljati struktur, ki bi pomenile dinamično povečevanje oziroma zmanjševanje potreb po pomnilniku. Prav tako mora imeti vsako opravilo, ki teče na takšnem sistemu pomnilnik v enem samem kosu (med začetnim in končnim fizičnim naslovom). Če imamo veliko opravil, ki tečejo na takšnem sistemu se lahko zgodi, da določenemu opravilu ne moremo zagotoviti pomnilnika. Poleg tega tak operacijski sistem ne vsebuje varovanja pomnilnika, ki ga zaseda določeno opravilo. Tako moramo sami zagotoviti, da opravilo dostopa samo do dela pomnilnika, ki mu je namenjen.

4.1.2 Realnočasovni operacijski sistemi

Realnočasovni operacijski sistem (RTOS) je operacijski sistem, za katerega je značilno, da se mora na dogodke odzivati pravočasno, oziroma da so določeni

intervali, v katerih morajo biti začeta opravila končana [30]. Pojem dejanskega časa je vezan na proces, v katerega je dani OS vključen. Pri jadralni navigacijski napravi takšen čas merimo v milisekundah.

Glavna značilnost RTOS je sposobnost preklapljanja med opravili, ki čakajo na procesorski čas. Povdarek je na algoritmih in implementaciji le teh, ki kakor je to le mogoče, omogočajo hitro in učinko preklapljanje med procesi. RTOS je zato uporaben predvsem tam, kjer je pomembno, kako hitro in s kakšno gotovostjo se lahko odzovemo na zahtevo, kot pa po svojih sposobnostih opraviti čim več dela v nekem času. Pravi RTOS zagotavljajo deterministično izvajanje opravil.

Preklapljanje med opravili poteka po navadi na podlagi pomembnosti opravila. Vsakemu opravilu lahko dodamo pomembnost. Upravljalnik opravil tako preklopi na novo opravilo šele, ko prejme zahtevo po procesiranju od opravila z višjo prioriteto. Druga možnost pa je, da vsakemu opravilu damo na voljo fiksno določen čas izvajanja. Ko čas poteče, po krožnem sistemu poiščemo naslednje opravilo, ki je na vrsti za izvajanje.

Sami smo si pogledali odprtokodno implementacijo realnočasovnega operacijskega sistema imenovano Free RTOS [19]. Gre za enega najpopularnejših realnočasovnih operacijskih sistemov, ki je bil zasnovan, da lahko teče na mikrokontrolerjih. Free RTOS je izredno preprost sistem, ki zavzema zelo malo prostora. Njegovo izvajanje je hitro. Ker je bil v zasnovan za izvajajnje na mikrokontrolerjih, zagotavlja zgolj najnujnejša orodja za gradnjo sistema. To je v našem primeru povsem zadosti, saj omogoča razvrščanje med opravili, medopravilno komunikacijo, štetje časa in sinhronizacijo. Vse ostale lastnosti pa lahko kasneje po potrebi vključimo kot dodatke. Pravzaprav predstavlja Free RTOS zgolj jedro nekega klasičnega RTOS.

Free RTOS je izredno enostaven za uporabo. Vse kar je potrebno storiti, da ga začnemo uporabljati je, da nekaj osnovnih datotek, ki vsebujejo programsko kodo vključimo v projekt. Koda je napisana v jeziku C in se nahaja v 3 datotekah. Poleg tega potrebujemo datoteko, ki je posebej prilagojena za tip mikrokontrolerja, ki ga uporabljamo. Free RTOS so podprli vsi večji

proizvajalci mikrokontrolerjev, tako da težav s tem ni. Poleg tega potrebujemo še konfiguracijsko datoteko. V njej nastavimo vrednosti določenih spremenljivk. Predvsem v zvezi z velikostjo pomnilnika in časi preklapljanja med opravili.

Opravilo, ki ga želimo poganjati na Free RTOS moramo napisati kot neskončno zanko. V njej napišemo del programa, ki predstavlja neko opravilo. Takšno opravilo nato dodamo na seznam opravil. Dodamo mu še naziv, velikost pomnilnika, ki mu je na voljo ter opcijske parametre. Najpomembnjša stvar je prioriteta opravila. Opravilom z nižjo prioriteto dodeljujemo nižje vrednosti, tistim z višjo pa višje. Najvišja vrednost je del konfiguracijske datoteke. Več opravil lahko ima tudi isto prioriteto. V takem primeru se bodo, če bodo seveda izpolnjenji vsi pogoji za njihovo izvajanje, izvajala istočasno v izmenjajočih časovnih intervalih. Ko smo dodali vsa potrebna opravila, ki sestavljajo našo aplikacijo, poženemo zgolj še organizatorja opravil.

Free RTOS poleg naših opravil vedno pripravi privzeto opravilo. To je namenjeno temu, da lahko organizator opravil vedno najde neko opravilo, ki mu lahko nameni procesorski čas. Kadar ni na voljo nobenega uporabniškega opravila, Free RTOS začne z izvajanjem privzetega opravila. V resnici se to opravilo ne izvaja, kar pomeni, da lahko v tem času procesor miruje. To pomeni zmanjšanje porabe energije. Pri gradnji naprave, ki se napaja zgolj na vgrajeno baterijo, predstavlja to pomembno lastnost. Takšna implementacija opravila, ki omogoča mirovanje procesorja pa ni optimalna, saj se mora vsake toliko časa procesor zbuditi in preveriti, ali že obstaja kakšno uporabniško opravilo, ki je pripravljeno za izvajanje.

4.1.3 Sistemi v neskončni zanki

Gre za najpreprostejši način implementacije programske opreme na mikrokontrolerjih. Tak sistem je tipično razdeljen na dva dela. V prvem delu se izvedejo vsi potrebni programski klici za inicializacijo vseh naprav in senzorjev, ki jih bomo uporabili. Izvede se inicializacija pomnilnika in vzpostavijo nastavitve. V pomnilnik preberemo program. Ko je praktično vse pripra-

vljeno za izvajanje, pa program preide v tako imenovano neskončno zanko. Gre za zanko, katere pogoj za vrnitev vanjo je vedno resničen.

V takšni zanki začnemo z implementacijo programskih elementov naše naprave. Tipično je prvi del takšne zanke sestavljen iz enega ali več odločitvenih dreves (switch-case), v katerih nato izvajamo posamezne dele aplikacije. Razbitje odločitvenega drevesa na prvem nivoju bi lahko pretvorili v posamična opravila.

V sistemu, ki teče v neskončni zanki moramo sami zagotoviti, da se aplikacija odziva na uporabniške zahteve, da imajo deli aplikacije dovolj časa za izvajanje svojih nalog in da pri tem ne ovirajo izvajanja drugih nalog.

Takšni sistemi so primerni predvsem za manjše projekte in jih kasneje precej težje povečujemo. Resno težavo predstavlja tudi odzivanje na prekinitvene zahteve, saj komunikacija med deli aplikacije poteka z globalnimi spremenljivkami. Prekinitvena zahteva lahko v času, ko je del apliakcije še uporabljal stare vrednosti teh spremenljivk, le-te spremeni. Po vrnitvi iz prekinitvene zahteve se tako aplikacija znajde v koruptiranem stanju. Z mehanizmi zaklepanja lahko to nevarnost sicer odpravimo.

4.1.4 Primerjava operacijskih sistemov in izbira ustreznega za našo implementacijo

Če spregledamo pomanjkljivosti s pomnilnikom, je uClinux spodoben operacijski sistem. Vsebuje ogromno mehanizmov, ki poenostavijo začetek razvoja naprave. A ravno dejstvo, da vsebuje vse v enem, predstavlja težavo pri gradnji naprave na omejeni strojni opremi. Zagonski čas, preden lahko poženemo aplikacijo, je tipično večji kot 5 sekund. Mikrokontrolerji kljub napredku niso bili razviti, da bi na njih tekli klasični operacijski sistemi, čeprav je to mogoče.

Veliko boljša izbira je Free RTOS. Free RTOS je mnogo preprostejši, a vseeno vsebuje vse potrebne mehanizme za hiter razvoj naprave. Je robusten, brez zagonskega časa in kar je najpomembneje, zagotavlja odzivnost na dogodke. Slednje je pomembno, ko gradimo napravo, ki komunicira z ogro-

mno podsistemi. Nekaj teh zahteva odzive v naprej predvidenih časovnih intervalih. To z uporabo klasičnega operacijskega sistema ne moremo doseči.

Izredno hitri za začetek razvoja so tudi sistemi, ki tečejo v neskončni zanki. Težava teh sistemov pa je kasnejše večanje kompleksnosti sistema. Razvijalec mora tako sam paziti, da bodo vsi deli aplikacije deležni dovolj procesorskega časa in predvsem v primeru gradnje letalske navigacijske naprave, da se bodo deli aplikacije pravočasno, to je v realnem času, odzvali na dogodke. To so lahko uporabniški vnosi ali pa komunikacija na enem izmed senzorjev, komunikacijski poti itd.

V našem primeru smo se tako odločili uporabiti Free RTOS. Gre za najpreprostejšo implementacijo RTOS, ki pa vsebuje vse potrebno za našo napravo. Omogoča nam, da preprosteje implementiramo predvsem tiste dele aplikacije, kjer moramo zagotoviti odziv na spremembe v okolju. Tu gre predvsem za pravočasno branje znakov na komunikacijskih poteh.

4.2 Implementacija specifičnih metod uporabnih v jadralnem letalstvu

4.2.1 Zapisovalnik leta IGC

IGC je kratica za mednarodno jadralsko komisijo. Sestavljena je iz predstavnikov članov nacionalnih zvez in je vrhovno telo v svetu športnega jadralnega letenja. Komisija določa športna pravila in kodekse. S prihodom navigacijskih naprav GNSS je komisija ustanovila sekcijo za nadzor in potrjevanje zapisovalnikov leta [9].

Zapisovalnik leta je najpomemnejša funkcija vsakega modernega jadralnega inštrumenta. Poznamo tri nivoje zapisovalnikov leta. V naši napravi smo implementirali najvišji nivo, ki zagotavlja da takšen zapisovalnik leta zapisuje lete, ki jih lahko piloti nato uporabijo za dokazovanje vseh dosežkov vključno s svetovnimi rekordi. Implementacija zapisovalnika leta na najvišjem nivoju zahteva, da smo se držali pravil predpisanih s strani IGC.

Najpomemnejša stvar pri zapisovalniku leta je, da uporabnik po končanem letu oziroma že med samim letom ne more vplivati na vsebino datoteke, ki na koncu predstavlja zapis leta. Ohišje naprave mora biti zasnovano tako, da onemogoča uporabniku dostop do mikroprocesorja, sprejemnika GNSS ali višinskega senzorja. V primeru, da uporabnik odpre napravo, pa se mora to nedvoumno zaznati. To lahko rešimo s stikali, ki so ob zaprti napravi stisnjena, če pa nekdo napravo odpre pa se razklenejo. Zapisovalnik leta let zapiše v tekstovno datoteko, katere vsebino bi lahko uporabnik poneveril. Da se to ne more zgoditi, je IGC predvidel standard podpisovanja datoteke. Zaključek datoteke vsebuje podpis, ki ga je zapisovalnik leta opravil z unikatnim privatnim ključem. Kakršno koli spreminjanje vsebine datoteke, bo posledično pomenilo, da je podpis neveljaven.

IGC datoteke je sestavljena iz glave, podatkov o letu in že zgoraj omenjnega podpisa. V glavi so podatki o zapisovalniku leta, imenu pilota in kopilota, podatki o letalu in podatki o deklarirani nalogi. Vse te podatke mora zapisovalnik leta zapisati v datoteko pred poletom. Vsakršno spreminjanje teh podatkov po tem, ko se let prične je nedovoljeno. Ko naprava zazna pogoje letenja, začne z zapisovanjem podatkov. Med letom zapisovalnik v določenih časovnih intervalih zapisuje podatke o poziciji GPS, višini GPS, višini pridobljeni iz senzorja višine in stopnji hrupa v okolici. Časovni interval je uporabiško nastavljiv in je tipično med 1 in 5 sekund. Hrup v okolici zaznavamo z mikrofonom. V primeru, da se naprava uporablja na jadralnem letalu z motorjem, mora zapisovalnik leta nedvoumno pokazati, kdaj je motor letala deloval. Ko pilot uporabi motor, da se izogne izven letališkemu pristanku, se njegov del leta, ki je veljaven za tekmovanje zaključi. Zapisovalnik leta mora zapisovati podatke vse do pristanka. Po pristanku zapisuje podatke še vsaj 2 minuti, da lahko sodniki nedvoumno razločijo fazo letenja od pristanka oziroma poleta. Ko se letenje zaključi. zapisovalnik leta podpiše nastalo datoteko in s tem prepreči morebitno spreminjanje vsebine.

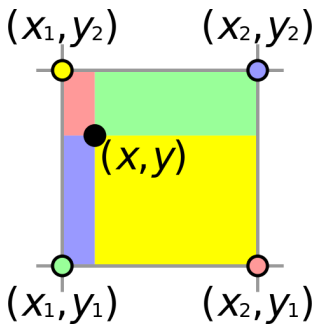
4.2.2 Prikaz zračnih prostorov

Pilot jadralnega letala lahko letalo upravlja samo v določenih zračnih prostorih. Za nekatere prostore potrebuje posebno dovoljenje kontrole zračnega prometa. Obstajajo pa tudi zračni prostori, kjer je letenje povsem prepoveano.

Zračni prostor je tridimenzionalno območje v zraku. Meje območja so podane z množico točk, ki so opisane kot koordinate v koordinatne sistemu Zemlje. Poleg točk je zračni prostor definiran s spodnjo in zgornjo višino prostora. Višine so lahko podane kot višine nad srednjim nivojem morja, višine nad zračno ploskvijo 1013 hPa (FL) ali pa kot višine nad terenom. V primeru, da je višina prostora podana kot višina nad terenom, potrebujemo za določanje višine zračnega prostora v neki točki tudi nadmorsko višino tiste točke na Zemlji. V našem primeru smo podatke nadmorskih višin pridobili iz brezplačne baze objavljene na spletni strani ameriške vesoljske agencije NASA [22].

NASA je na spletu objavila podatkovno bazo višin. Gre za mednarodni projekt imenovan SRTM, kjer je NASA s pomočjo satelita, ki ga je v vesolje ponesel raketoplan Endeavour, na enajstdnevnem potovanju okoli zemeljske oble skenirala površje Zemlje. Pri tem so uporabili tehnologijo InSAR. Dobljeni podatki so na voljo kot območja, ki predstavljajo eno stopinjo geografske širine in eno stopinjo geografske dolžine. Prosto izdani podatki za celoten svet so na voljo za točke, ki so med seboj oddaljene 90 metrov.

Da bi ugotovili dejansko višino za določeno točko, smo implementirali bilinearno interpolacijo [4]. Bilinearna interpolacija je pravzaprav razširitev linearne interpolacije na interpolacijo dveh spremenljivk v dveh pravokotnih oseh. Osnovna ideja je, da opravimo linearno interpolacijo v vaski smeri posebej. Vsaka posamezna vrednost je tako resnično linearna, rezultat kot celota pa v omenjenem prostoru pa ni, ampak je kvadratičen. Za zahtevano pozicijo najprej poiščemo točke, ki jih imamo v naši bazi in za katere poznamo višino. Točke izberemo tako, da obdajajo iskano točko, kot je prikazno na sliki 4.1.



Slika 4.1: Izbira ustreznih izhodišč (x1, x2, y1, y2) za iskanje zahtevane vrednosti s pomočjo bilinearne interpolacije za izbrano točko (x, y).

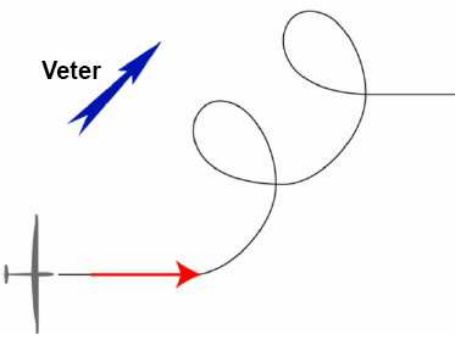
Pomemno je, da zračni prostor prikažemo na zaslonu tako, kot so zračni prostori prikazani na fizičnih letalskih kartah. Na letalskih kartah lahko vidimo, da so zračni prostori prikazani kot območja katerih meja je prekinjena črta. Določena območja, kjer letenje ni dovoljeno oziroma je za letenje v tem območju potrebna odobritev kontrole letenja so na fizičnih letalskih kartah tudi pobarvana. Tipično z modro ali rdečo transparento barvo, tako da je vsebina pod območjem vseeno vidna.

4.2.3 Izračun vetra

Eden izmed pomembnejših podatkov, ki jih lahko pilot dobi samo z uporabo modernih navigacijskih naprav je hitrost in smer vetra. Na podlagi teh podatkov se odloča, kje iskati dviganja ter kako načrtovati svoj let, da bo ta varen in obenem učinkovit. Hitrost in smer vetra odločilno vplivata na dolet jadralnega leta, kar bomo videli v poglavju 4.2.4.

Veter lahko izračunamo na več različnih načinov. Naša naprava je opremljena samo s sprejemnikom GPS, kar nam omogoča zgolj izračun vektorja vetra v času, ko jadralno letalo kroži. Jadralna letala pogosto krožijo, saj je to eden izmed načinov, kako jadralno letalo pridobi višino, ki jo lahko nato uporabi za ravni let. To pomeni, da imamo dovolj priložnosti, da izračunamo smer in jakost vetra. Ob predpostavki, da se hitrost in smer vetra v prostoru

spreminja zvezno in so te spremembe relativno majhne glede na območje, kjer operira jadralno letalo med dvemi zaporednimi dviganji, lahko ugotovimo, da lahko dobljen veter uporabimo pri vseh ostalih izračunih, ne da bi pri tem zagrešili večjo napako.



Slika 4.2: Pot letala glede na tla pri kroženju, ko piha veter.

Pri kroženju jadralnega letala lahko opazujemo dva učinka, ki jih ima veter na hitrost in trajektorijo leta jadralnega letala glede na Zemljo. Posledično lahko veter izračunamo na dva različna načina, uporabniku pa dopustimo, da izbere metodo, ki jo želi uporabiti.

Razlika hitrosti

Prvi način, da pridobimo hitrost vetra med letom je, da opazujemo, kako se spreminja hitrost jadralnega letala glede na tla (GS), med tem ko jadralno letalo kroži. Opazimo lahko, da je hitrost glede na tla največja takrat, ko letalo leti z vetrom in najmanjša takrat, ko letalo leti proti vetru. Razlika, ki je nastala, je posledica vetra, saj letalo glede na zrak leti s konstantno hitrostjo. Sama vrednost hitrosti glede na zrak je za izračun nepomembna. Pomembno je samo ugotoviti razliko med najmanjšo in največjo vrednostjo hitrosti glede na tla. Dobljena razlika predstavlja dvojno hitrost vetra, zato jo moramo za končno hitrost deliti z dva. Smer vetra pa je očitno smer letenja, ko smo izmerili največjo hitrost glede na tla.

Tako lahko v vsakem zavoju izračunamo hitrost vetra. Dobljene hitrosti

nato filtriramo, saj so podvržene letenju z nekonstantno hitrostjo glede na zrak. Po nekaj zavojih imamo tako dober izračun vektorja vetra, ki ga lahko uporabimo za ostale izračune.

Zanos

Drugi način, da pridobimo hitrost vetra med letom je, da opazujemo trajektorijo letala med kroženjem. Kljub temu, da letalo med kroženjem leti po krožnici glede na zrak, se zaradi vetra trajektorija glede na Zemljo spremeni v spiralo. To je vidno na sliki4.2. Pride do tako imenovanega zanosa vetra. Da bi izračunali veter, moramo pravzaprav izračunati zanos v odvisnosti od časa. Zanos izračunamo tako, da po vsakem končanem zavoju poiščemo središče tega zavoja. Ko imamo vsaj dva popolna zavoja lahko povežemo dobljeni središči. Smer vektorja, ki ga dobimo je tudi smer vetra. Ob vsakem zaključenem popolnem zavoju si zapomnimo tudi trenuten čas. Na ta način lahko iz razdalj med središči in razlik v času izračunamo hitrost zanosa, kar predstavlja tudi hitrost vetra.

4.2.4 Izračun doleta

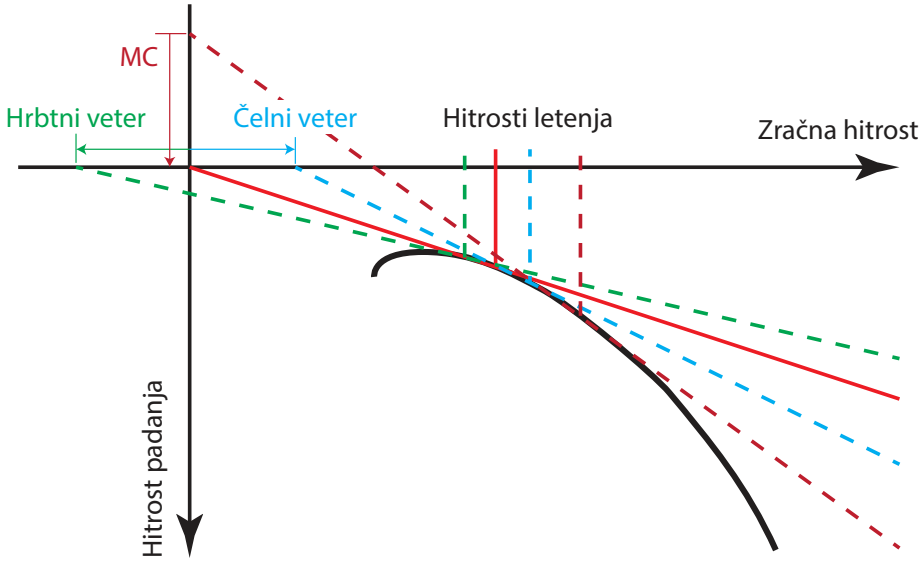
Izračun doleta je eden izmed glavnih podatkov, ki jih lahko da sodobna navigacijska naprava pilotu. Dolet pomeni ali je pilot v trenutnih razmerah zmožen doseči izbrano točko. Seveda še tako dober izračun ne more predvideti vremenske situacije skozi katero bo pilot letel na poti do točke, a je v praksi z ustrezno nastavljenimi parametri dovolj zanesljiv. Dolet največkrat prikažemo kot relativno višino, ki jo bo letalo imelo nad izbrano točko, ko doseže izbrano točko, ne da bi pri tem pridobil dodatno energijo iz okolja. Osnova za izračun doleta je polara jadralnega letala, ki je podrobneje predstavljena nekoliko kasneje.

Osnovno polaro prilagodimo glede na trenutno maso jadralnega letala. Jadralno letalo lahko med letom izpusti vodni balast, ki se nahaja v krilih in s tem spremeni svojo polaro. Pri dani polari nato izračunamo najboljšo fineso [5].

Nato izračunamo razdaljo od naše trenutne pozicije do izbrane točke. S pomočjo najboljšega drsnega razmerja izračunamo potrebno višino, da pridemo to izbrane točke, torej kako visoko moramo začeti drsni let, da bomo na koncu premagali zastavljeno razdaljo. Dobljeni višini odštejemo višinsko razliko med našo trenutno višino in višino izbrane točke. Dobljena višina nam pove relativno višino na katero bomo prileteli nad izbrano točko. Če je vrednost negativna bi to pomenilo, da točke ne moremo doseči, saj bi prišli do točke pod površjem Zemlje. Če je vrednost pozitivna, pa let lahko nadaljujemo, saj bomo izbrano točko lahko dosegli. Seveda ta izračun predvideva, da bomo ves čas leteli s hitrostjo najboljšega drsnega razmerja in ob enem leteli v mirni zračni masi. Ker pa takšno letenje na eni strani ni najhitrejše, na drugi strani pa nikoli ne moremo leteti v popolnoma mirni zračni masi, to je v zraku kjer ni nobenega vertikalnega oziroma horizontalnega gibanja zraka, je Paul MacCready (MC) iznašel kako lahko na preprost način izračun doleta dopolnimo [28]. Tako dopolnjen izračun upošteva vremensko situacijo v zraku in pilotu poda podatek s kakšno hitrostjo naj leti proti izbrani točki, da bo ta let najučinkovitejši. S tem je mišljeno, da bo povprečna hitrost letenja najvišja. To je še posebej pomemno pri tekmovalnem jadralnem letenju.

Da dobimo drsno razmerje in hitrost s katero moramo leteti, je potrebno na podlagi teorije, ki jo je razvil MacCready, spremeniti izhodišče koordinatnega sistema, s katerim smo opisali polaro letala. V primeru, da pričakujemo vreme, kjer bodo povprečni vzgorniki dosegli hitrosti 2m/s, lahko posledično pričakujemo tudi večja padanja letala, ko letalo leti med dvema vzgornikoma. Izhodišče koordinatnega sistema tako prestavimo na 2m/s. Iz novega izhodišča potegnemo tangento na polaro. Točka, ki jo dobimo, predstavlja drsno razmerje, ki ga lahko pričakujemo ob letenju v takih vremenskih pogojih in s takšno hitrostjo.

Na dolet pa vpliva tudi veter. Zanima nas predvsem čelna oziroma hrbtna komponenta vetra v smeri leta proti izbrani točki. Sprememba koordinatnega izgodišča v primeru hrbtnega oziroma čelnega vetra je prikazana na sliki 4.3.



Slika 4.3: Prikaz hitrosti najboljšega drsnega razmerja v primeru, ko piha čelni oziroma hrbtni veter. MC prikazuje prestavljeno izhodišče koordinatnega sistema zaradi povprečnih vzgornikov.

4.2.4.1 Polara letala

Polara jadralnega letala opisuje sposobnosti jadralnega letala. V osnovi gre za razmerje horizontalne hitrosti letala skozi zrak in padanja oziroma vertikalne hitrosti letala. Proizvajalci jadralnih letal polaro objavijo v priročniku jadralnega letala. Do polare pridemo z izredno natančnimi meritvami, ki morajo potekati v mirni atmosferi. Z jadralnim letalom letimo v različnih hitrostnih režimih in ob tem beležimo padanje oziroma vertikalno hitrost jadralnega letala. Da bi zagotovili pristnost takšnih meritev, vedno poleg leti jadralno letalo z znano in večkrat kalibrirano polaro. Tako lahko potrdimo, da so izmerjeni parametri pristni in dejansko kažejo lastnost jadralnega letala, ne pa vplivov iz okolja. Dobljene rezultate nato povežemo. Dobljeni

krivulji pravimo polara jadralnega letala.

So pa dobljeni rezultati zelo odvisni od tega, kakšna je bila masa jadralnega letala med testnim letom. Zato proizvajalci ponavadi opravijo več meritev pri različnih masah jadralnega letala. Tipično pri praznem jadralnem letalu v katerem sedi zgolj pilot. Nato dodamo vodni balast, tako da pridemo na neko srednjo vrednost med minimalno in maksimalno vzletno maso. Na koncu pa opravimo še let z maksimalno dovoljeno vzletno maso jadralnega letala. Ker se površina kril ob spremembi mase ne spremeni, se obremenitev letala podaja kot krilna obremenitev. Krilna obremenitev je razmerje med maso letala in površino kril. Zanimivo je, da se oblika same krivulje pri različnih krilnih obremenitvah ne spremeni. Se pa sama krivulja pri večjih krilnih obremenitvah premakne vzdolž osi, na katero nanašamo horizontalno hitrost letala. Ta ugotovitev je precej ugodna, saj si lahko v naši bazi shranimo samo eno polaro za vsako jadralno letalo. In če poznamo referenčno krilno obremenitev, torej krilno obremenitev v času, ko smo izmerili shranjeno polaro, potem lahko iz podatkov o trenutni masi letala preračunamo trenutno polaro.

$$E = \frac{Y}{X} = \frac{C_y}{C_x} = \frac{L}{H} \tag{4.1}$$

kjer je:

E - drsno število jadralnega letala

Y - vzgon letala (N)

X - upor letala (N)

C_y - vzgonski količnik krila

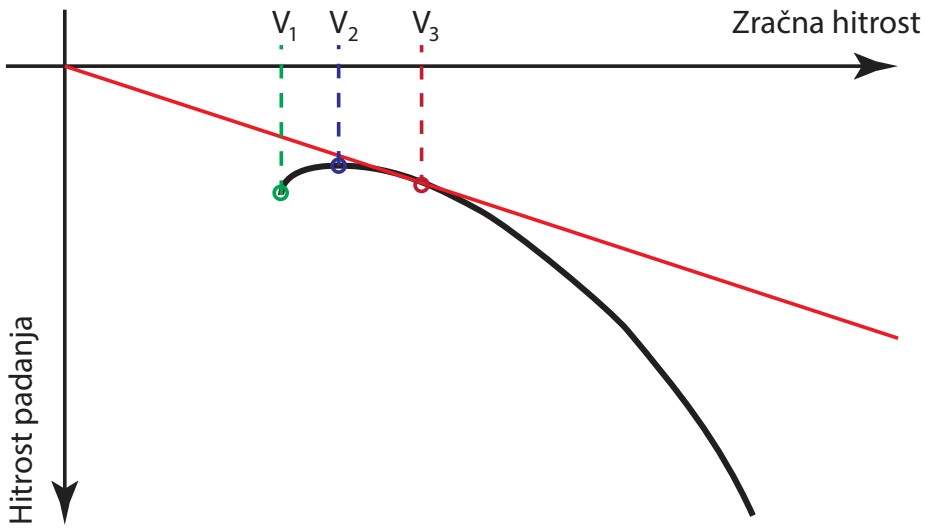
C_x - količnik upora letala

L - dolet jadralnega letala (m)

H- začetna višina (m)

Polara vsebuje tudi nekaj zanimivih točk, ki ji uporabljamo pri različnih izračunih. Najpomembnejša je zagotovo finesa oziroma drsno razmerje jadralnega letala. Označimo jo z E. Gre za točko, ki nam pove pri kateri

horizontalni hitrosti bo jadralno letalo letelo najdlje iz dane višine. Finesa jadralnega letala je torej razmerje med preleteno razdaljo in izgubljeno višino [13]. Podamo jo lahko tudi na drug način 4.1. Fineso določimo tako, da iz izhodišča koordinatnega sistem potegnemo tangento na polaro.



Slika 4.4: Prikaz polare jadralnega letala. V_1 je najmanjša hitrost, V_2 je hitrost najmanjšega padanja, V_3 je hitrost najboljšega drsnega razmerja.

Kot smo lahko videli zgoraj, je polara krivulja, ki je nastala na podlagi meritev v naravi in je zato podvržena vplivom narave. Kljub navidezni kompleksnosti pa se je tudi zaradi omejene procesorske moči in pomnilnika, za shranjevanje polar v digitalni obliki, uveljavila aproksimacija s pomočjo polinoma 2 stopnje oziroma tako imenovane kvadratne enačbe. Seveda bi za natančnost lahko uporabili polinome višjih stopenj, a se je v praksi izkazalo, da je aproksimacija polar s polinomom druge stopnje povsem zadosti. Kljub temu, da se polinom v zgornjem in spodnjem delu hitrostnega območja

jadralnega letala ne prilagaja polari najbolje, je prileganje pravi polari v srednjem delu precejšnje, to pa je tudi območje, kjer jadralna letala večinoma letijo.

Poglavje 5

Grafično uporabniški vmesnik

Grafični uporabniški vmesnik (GUI) omogoča komunikacijo med uporabnikom in napravo s pomočjo grafičnih elementov prikazanih na zaslonu. Pred začetkom uporabe grafičnih uporabniških vmesnikov je komunikacija z računalnikom potekala s pomočjo vmesnika z ukazno vrstico (CLI). Pri tem je uporabnik s pomočjo tipkovnice vnašal ustrezne ukaze, rezultat le teh pa se je izpisal na zaslonu. Tak vmesnik pa je neprimeren za širšo uporabo. Grafični uporabniški vmesniki so veliko bolj intuitivni za uporabo. Ljudje si hitreje zapomnimo oblike kot pa zgolj besedilo. Takšni vmesniki so zato na vseh modernih napravah namenjenih splošni uporabi [14].

Obstaja kar nekaj načinov, kako lahko uporabnik upravlja z grafičnim uporabniškim vmesnikom. V začetku je večina vmesnikov podpirala interakcijo s pomočjo miške in tipkovnice. Sami pa se bomo osredotočili zgolj na interakcijo s pomočjo zaslonov občutljivih na dotik. S pojavom pametnih telefonov in tablic je takšna interakcija postala zelo pogosta. Gre za izredno neposredno interakcijo z GUI, saj se uporabnik s konico prsta svoje roke praktično dotika elementov GUI.

5.1 Osnove gradnje učinkovitih uporabniških vmesnikov

Izgradnja učinkovitega grafičnega uporabniškega vmesnika ni samoumevna. Potrebuje veliko načrtovanja. Predvsem kadar moramo na majhni površini zaslona prikazati veliko informaciji. Prikaz informacij mora tako poleg samih informacij zagotoviti, da prikaz ne vpliva na prikaz drugih informacij in da je prikaz informacij konsistente. Konsisteno mora delovati tudi odzivanje na uporabniške vnose preko zaslona občutljivega na dotik. V osnovi pa je bistvo učinkovitega uporabniškega vmesnika, da dobro deluje v praksi, je enostaven za učenje in ga ljudje z veseljem uporabljajo [27].

Moderni uporabniški vmesnki na mobilnih napravh in navigacijskih napravh v letalstvu upoštevajo te smernice. Zato si bomo v nadaljevanju ogledali, kako so zgrajeni takšni GUI. Znanje pa uporabili za gradnjo naše GUI. Videli bomo da so GUI v napravah namenjenih letalstvu precej bolj pusti in upoštevajo pravilo, da s čim manj porabljenega črnila prikažemo vse informacije. Na zaslonih tako ne najdemo nobenih nepotrebnih elementov. Pri GUI na mobilnih napravah pa se tu in tam vendarle pojavijo elementi, ki so na zaslonu zgolj zaradi estetike in ne nosijo prav nobene dodatne informacije.

Sami se bomo poskušali držati načela, ki ga je razvil Tufte [35]. Uvedel je razmerje med črnilom porabljenim za prikaz podatkov in celotnim porabljenim črnilom. V idealnih pogojih je to razmerje 1.0. Vsak nepotreben element poslabša to razmerje. Dobra grafika tako vsebuje zgolj črnilo porabljeno za prikaz podatkov, vse ostalo pa zavrže. To drži tudi za prikaz elementov na zaslonu.

5.2 Primeri dobre prakse v letalstvu

Pred začetkom gradnje našega grafičnega uporabniškega vmesnika, smo si ogledali kakšni so trenutni vmesniki na napravah, ki so namenjene navigaciji v letalstvu in so trenutno na voljo na tržišču. Na trgu trenutno ne ob-

staja nobena naprava namenjena jadralnemu letenju z GUI, ki bi deloval z večdotičnim zaslonom občutljivim na dotik. Takšni zasloni se sicer uporabljajo v vgrajenih navigacijskih napravah namenjenih večjim motornim letalom. Zato smo si ogledali kakšni so ti vmesniki, ter vmesniki na vgrajenih modernih jadralnih navigacijskih napravh.

Ko govorimo o gradnji naprav za splošno letalstvo (ang. General aviation), obstajajo za vsak del razvoja takšne naprave določena pravila. Ta pravila postavljajo organizacije, ki skrbija za certificiranje takšnih naprav. V Evropi za to skrbi EASA v Združenih državah Amerike pa FAA. Te organizacije so tako za gradnjo modernih digitalnih inštrumentov predvidele tudi pravila za gradnjo grafičnih uporabniških vmesnikov [8].

V dokumentih, ki so na voljo vidimo, da je povdarek pri gradnji uporabniškega vmesnika v tem, da je takšen vmesnik jasen, elementi morajo biti nedvomljivi. Precej pozornosti je namenjene izbiri barv. V letalsvu imajo določene barve prav poseben pomen, zato jih ne smemo uporabljati nenamensko [3]. Najpomembneje je, da nenavadne situacija oziroma meritve, ki kažejo odstopanje od normalnega delovanja letala nedvoumno prikažemo na zaslonu. Tako imajo rdeča, rumena in oranžna barva prav posebno mesto. Vse nenavadne situacija morajo pasti v eno od treh skupin obvestil, ki jih uporabnik vidi na zaslonu. Opozorila, ki kažejo največja odstopanja od normalnega delovanja morajo biti prikazana rdeče. Gre za primere, ko bi nadaljevanje letenja v takšnih pogojih lahko pripeljalo do nesreče. Naslednja so obvestila o nevarnosti. Ta morajo biti obarvana oranžno ali rumeno. Najmilejša stopnja obvestil, ki uporabniku zgolj svetujejo, pa so lahko obarvana v poljubni barvi, vendar ne v rdeči in zeleni. Zelena se v letalstvu smatra kot barva, ki kaže, da so parametri znotraj predvidenih območij. Kadar uporabnik prejme obvestilo, se to zgodi zato, ker eden ali več parametrov niso znotraj predvidenega območja. Poleg barve pa morajo vsa obvestila oziroma opozorila vsebovati vsaj en dodaten vizualni element, da se razlikujejo od normalne vsebine in pritegnejo pozornost uporabnika. To je lahko velikost pisave, masten tisk, pozicija na zasloni, utripanje, itd. Naprava mora upo-

rabnikovo pozornost za opozorila pridobiti z najmanj dvema dražljajema, ki delujeta na različna čutila. Najpogosteje tako opozorilom na zaslonu sledijo zvočni ali vibracijski efekti.

Lahko pa se rezervirane barve (rdeča, rumena, zelena, oranžna) uporabljajo za prikaz informacij na zaslonu, ki prikazujejo stanje na mapi v okolici letala. To so lahko podatki o višini terena, padavinah, poti, ki jo je že naredilo letalo. Gre torej za infomacije, ki so prikazane kot plasti na mapi in jih lahko po potrebi uporabnik tudi izklopi.

Ob analizi grafičnih uporabniških vmesnikov uporabljenih v letalstvu lahko ugotovimo, da so vsi zgrajeni tako, da je ozadje prikazano črno. To je seveda logično, saj ozadje ne nosi nobene informacije, mi pa želimo da so dejanske informacije prikazane čim bolj kontrastno. Raznobarvnih ozadij tako ne srečamo, razen ko ta prikazujejo teren nad katerim letimo. Lahko bi bila ozadja tudi bele barve, a bi se informacije lahko ob močni direktni sončni svetlobi izgubile v ozadju. V splošnem so grafični elementi sestavljeni tako, da omogočajo nedvoumno zaznavanje v vseh svetlobnih pogojih. Pri razvoju morajo tako upoštevati močno sonce s strani, spredaj skozi steklo, kar posledničnо pomeni odsevanje oblačil v zaslonu in letenje v slabih svetlobnih pogojih. Poleg tega so na letalu ves čas prisotne vibracije. Elementi so zato postavljeni tako, da je med njimi dovolj razmika. Skupine elementov, ki prikazujejo podobne informacije so na vseh zaslonskih maskah prikazane enako. To skrajša proces učenja in poveča učinkovitost prikaza informacij. Velikost pisave in ostalih grafičnih elementov pa je prilagojena predvideni oddaljenosti uporabnika od zaslona.

Prav tako dokumenti o gradnji grafičnih uporabniških vmesnikov priporočajo uporabo meril, trakov oziroma števcev za prikaz najpomembnejših numeričnih informacij. Gre pravzaprav za digitalen prikaz analognih naprav. Ugotovljeno je bilo, da uporabniki veliko lažje zaznajo dejansko numerično vrednost, predvsem pa lahko iz gibanja kazalca ugotovijo trend, torej kaj se z informacijo dogaja. Poleg tega lahko v takšnih trakovih oziroma števcih prikažemo normalna in nenormalna območja delovanja. Tak prikaz

Slika 5.1: Primer GUI na certificirani navigacijski napravi namenjeni vgradnji v motorna letala Splošnega letalstva. GUI upošteva vsa priporočila.

numerične informacije združuje več informacij in je boljši kot zgolj izpis števk na zaslonu.

Za boljše razumevanje kaj določeni podatki pomenijo, se uporablja standardna angleška letalska frazeologija. Večina podatkov je prikazana pod kraticami, ki so zbrane v dokumentu [15]. Za jasnejše podajanje informacij so poleg besedila včasih tudi ikone. Tudi oblika in postavitev ikon ni prepuščena posameznemu razvijalcu, ampak tudi za to obstajajo standardi [17].

Zgoraj opisana pravila in opažanja se nanašajo na grafične uporabniške vmesnike, ki jih najdemo v splošnem letalstvu. Poglejmo si še, ali podobna pravila upoštevajo tudi snovalci moderni jadralnih navigacijskih naprav. Ugotovimo lahko, da grafični uporabniški vmesniki skoraj povsem upoštevajo zgoraj opisana opravila. Največja odstopanja so pri uporabi barv,

Lastnost	**Barva**
Opozorila	rdeča
Prekoračitev normalnega območija delovanja	rdeča, rumena, oranžna
Obvestila, ne normalna	rumena, oranžna
Trakovi, števci, merila	bela
Zemlja	rjava
Nebo	modra
Vključeni parametri in normalno delovanje	zelena
Linije za razmejevanje elementov, neaktivna polja	rahlo siva

Tabela 5.1: Priporočene barve za različne vrste elementov

predvsem rabi rdeče in rumene. GUI na jadralnih napravah te barve uporabljajo tudi za izpis parametrov, ki so povsem znotraj mejnih območij. Primer GUI na moderni vgradni navigacijski napravi za jadralna letala lahko vidimo na sliki 5.2 Predvsem se veliko uporablja rumena barva za prikaz numeričnih vrednosti na črnem ozadju. Razlog za to je veliko kontrastno razmerje med črnim ozadjem in rumenim podatkom. Sicer pa je podobnost z GUI, ki jih najdemo na napravah v motornih letalih dobra iz povsem praktičnega vidika. Ogromno jadralnih pilotov v službenem času leti motorna letala. Upoštevanje istih pravil poenostavi prehod iz enega v drug sistem.

Ker pa na trgu ni nobene mobilne jadralne navigacijske naprave, ki bi za interakcijo uporabljala kapacitivni zaslon občutljiv na dotik, smo si v naslednjem poglavju ogledali primere dobre prakse grafičnih uporabniških vmesnikov na mobilnih napravah. Znanje, ki ga bomo dobili pri tem bomo združili s pravili za gradnjo GUI opisanih zgoraj. Tako bomo v nadaljevanju zgradili GUI za našo napravo.

Slika 5.2: Primer GUI na jadralni navigacijski napravi LX Zeus.

5.3 Primeri dobre prakse na mobilnih napravah

V tem poglavju bomo pogledali, kakšni so grafični uporabniški vmesniki na modernih mobilnih napravah, ki za interakcijo uporabljajo zaslon občutljiv na dotik. S pojmom moderne mobilne naprave se osredotočamo na mobilne telefone, ki so po velikosti precej podobni naši napravi. Ogledali si bomo lastnosti dveh operacijskih sistemov, ki tečeta na večini mobilnih naprav. Upoštevanje pravil, ki jih pri gradnji svojih GUI uporabljajo te naprave, bomo izkoritili pri gradnji GUI za našo napravo in tako poenostavili uporabo naše naprave.

Na večini naprav danes tečeta Android ali iOS operacijska sistema. Prvi

Slika 5.3: Primeri GUI na mobilnih napravah. Prvi na levi je iOS, vsi ostali pa predstavljajo Android operacijski sistem.

se nahaja v nekaj različicah, saj imajo prozivajalci OEM zaradi odprtokodnosti operacijskega sistema možnost prilagoditve GUI svojim potrebam [31]. Vseeno pa se za našo analizo razlike med različnimi verzijami Andriod operacijskega sistema preveč podrobne. Vsi sistemi upoštevajo ista pravila, ki jih je Google, kot snovalec operacijskega sistema združil v spletni zbirki imenovani Google Material Design [12].

Google Material Design je priporočilo za gradnjo vseh novih grafičnih uporabniških vmesnikov na kapacitivnih zaslonih občutljivih na dotik, ki omogočajo večprstno zaznavanje. Ni namenjen zgolj za gradnjo Android operacijskega sistema ali razvijalce aplikacij za ta operacijski sistem. Upoštevanje teh pravil poenostavi prehajanje uporabnikov iz ene na drugo napravo. Predvsem pa podaja smernice za nedvoumen prikaz informacij na zaslonu. Priporočila so osnovana na primerjavah med vlogami senc, ki so posledica osvetljevanja papirja, ki ga premikamo v tri dimenzionalnem prostoru. Tako je tudi v digitalnem svetu, kjer vsakemo elementu v GUI, ki ima že sam po

sebi širino in dolžino, dodamo podatke o tretji dimenziji. Os z gleda pravokotno iz zaslona. Vsak element je debel točno 1dp in ima eno vrednost na osi z. Temu rečemo višina. Z namišljenim osvetljevanjem takšnega prostora lahko zgradimo hierarhijo med elementi, ki je dobro vidna s sencami. Dejansko vsebino potem prikažemo na enem izmed elementov, kot bi s črnilom pisali po papirju. Za pravila med elementi veljajo isti odnosi kot med listi papirja. Ena pomembnejših posledic je, da dva elementa ne moreta zasedati istega prostora. Če se v dvo dimenzionalnem pogledu prekrivata, morata biti postavljena na različnih višinah.

Elementi se lahko premikajo po vseh oseh. Premikanje po osi x in y je ponavadi posledica animacij, premikanje po osi z pa posledica interakcije z uporabnikom. Elementi se lahko tudi združujejo, delijo in vrtijo a se ne morejo zvijati. Da se vzpostavi hierarhija med elementi na zaslonu, so snovalci Google Material Design predvideli privzete višine za vse osnovne elemente. Razvijalcem aplikacij niti ni potrebno skrbeti za te podrobnosti, saj ob uporabi privzetih gradnikov, prevzamejo tudi njihove lastnosti.

Tako Android kot tudi iOS operacijska sistema poskušata elemente na zaslonu animirati tako, da uporabnik izgubi občutek, da upravlja z navideznimi elementi. Elementi, ki se premikajo po zaslonu upoštevajo zakone fizike. Za primer si vzamimo premikanje po meniju, ki zaseda več prostora kot ga je na voljo na zaslonu. Da bi se premaknili na skrite dele menuja, moramo s prstom povleči tako, da trenutno vidni del prestavimo iz zaslona. Če to storimo počasi, bodo elementi skoraj v popolnosti sledili našim prstom. V primeru, da to storimo sunkovito se pod našimi prsti zgodi večji premik, kot smo ga sami opravili. A tudi ta animacija upošteva zakone fizike. Na začetku vsebina pospešuje, nato večji del časa potuje po zaslonu z konstantno hitrostju, na koncu pa se počasi upočasni zaradi navideznega trenja. Da animacije izgledajo naravno je pomemno, kako animacijo začnemo, ter kako jo končamo. Poleg tega moramo zagotoviti dovolj veliko osveževanje slike na zaslonu. Vse animacije na Androidu in iOSu so hitre in odločne, kar da uporabniku občutek, da sistem teče gladko. Animacije se zgodijo v povprečju

Slika 5.4: Shematičen prikaz gradnje GUI s pomočjo pravil opisanih v Google Material Design. Vrednosti v oklepajih predstavljajo navidezno oddaljenost od izhodišča osi z.

znotraj 300ms. Večje animacije se zgodijo v 375ms. Animacije elementov, ki pridejo na zaslon se zgodijo v 225ms. Animacije elementov, ki zapustijo zaslon se zgodijo v 195ms. Na sistemih ni nobenih animacij, katerih trajanje bi preseglo 400ms. Takšna dolžina že predstavlja občutek, da sistem ne teče gladko.

Naslednje področje pa je izbira barv oziroma barvnih kombinacij. Tukaj so razlike med operacijskimi sistemi precejšnje, a vsi vseeno upoštevajo določena pravila. Vsak GUI je sestavljen iz precej majhnega števila barv, kjer vsaka barva daje neko dodatno lastnost elementu, ki je z njo obarvan. Pri iOS veliko bolj intenzivno uporabljajo transparentnost elementov. Vsebina, ki se nahaja pod transparentnim elementom je popačena, a obrisi so vseeno vidni. Uporabnik, tako dobi občutek, kaj lahko pričakuje, ko se bo vrhnji element odstranil iz zaslona.

Tudi Apple je za svoj iOS izdal dokument iOS Human Interface Guidli-

nes [2]. Dokument je predvsem namenjen razvijalcem aplikacij, ki tečejo na operacijskem sistemu iOS. Razvijalci morajo priporočila iz dokumenta strogo upoštevati. Apple si pridržuje pravico, da zavrne aplikacijo, ki ne upošteva njihovih pravil gradnje GUI. Ker pa je večina aplikacij sestavljena iz privzetih gradnikov, je za upoštevanje pravil poskrbljeno kar v razvijalskih orodjih.

Poglejmo si še, kako se uporabniki premikajo po napravi, ki uporablja zaslon občutljiv na dotik. Večina interakcije je zelo intuitivna. Da izberemo določen element, preprosto kliknemo nanj s prstom. Če imamo seznam in se želimo premikati po njem to storimo tako, da s prstom vlečemo po zaslonu, tako da določeni elementi padejo iz zaslon, na drugi strani pa se pojavijo novi. Zaslon si tako lahko v večini primerov predstavljamo kot lupo, ki jo premikamo po papirju. Za to da vsebino povečamo ali pomanjšamo se uporablja dvoprstna kretnja. Približevanje dveh prstov zmanjša prikaz informacij na zaslonu, oddaljevanje dveh prstov pa prikaz informacij poveča. Informacije lahko tudi zasučemo, tako da dva prstav vrtimo po zaslonu. To so osnovne operacije, ki jih podpirajo vse mobilne naprave, ki uporabjajo večdotični zaslon. Vsak operacijski sitem pa podpira še nekaj drugih kretenj, ki predstavljajo bližnjice ali imajo za posledico kakšno specifično akcijo.

5.4 Implementacija uporabniškega vmesnika prilagojenega za zaslon občutljiv na dotik

Z znanjem, ki smo ga pridobili pri analizi GUI na mobilnih napravah, ki uporabljajo zaslon občutljiv na dotik in pravili, ki veljajo za GUI v letalskih napravah, smo se lotili gradnje GUI za našo napravo. Najprej smo zgradili osnovne elemente iz katerih smo nato sestavili celoten sistem. Preden smo začeli s postavljanjem elementov na zaslon, smo si ustvarili mrežo, ki je razdelila zaslon na enakomerne dele. Na ta način smo povečali preglednost zaslona in zagotovili dovoljšno medsebojno oddaljenost elementov.

Ozadje naprave, je temno. Zaradi tipa zaslona, ki ga naprava uporablja, barva ni povsem črna, a smo s testiranji zagotovili, da smo izbrali barvo s

katero lahko dosežemo velike kontraste z elementi, ki predstavljajo vsebino. Vse informacije so prikazane s svetlimi barvami. Pri določenih elementih barve ne uporabljamo striktno s pravili, ki veljajo za izbiro barv skladno s tabelo 5.1. Gre predvsem za prikaz podatkov, ki so specifični v svetu jadralnih letal. Ti podatki so v podobnih barvnih shemah prikazani tudi na modernih jadralnih navigacijskih napravah, zato smo tudi sami ohranili te lastnosti.

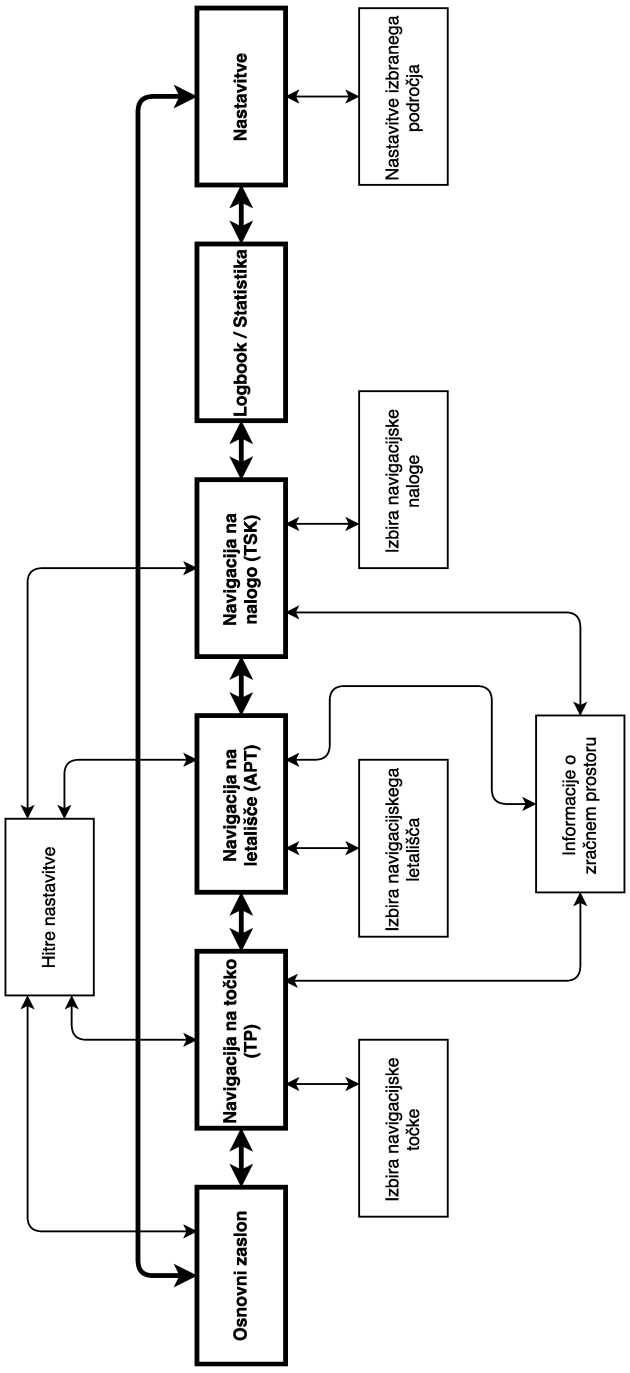
Ko smo imeli osnovne elemente pripravljene, smo se lotili implementacije posameznih zaslonskih mask. Preden pa se lotimo same implementacije, pa si moramo ustvariti osnovno predstavo, kako si posamezne zaslonske maske sledijo v sami napravi. Ko napravo prižgemo se na zaslonu prikaže Osnovni zaslon. Na istem nivoju imamo vsega skupaj 6 zaslonskih mask, ki so urejene v krožnem sistemu. S tem smo poenostavili prehajanje med zaslonskimi maskami in ohranili hierarhijo, ki jo najdemo na modernih jadralnih navigacijskih napravah. Iz teh osnovnih zaslonskih mask nato dostopamo do vseh ostalih. Osnovna shema zaslonskih mask je prikazana na sliki 5.5.

V naslednjih poglavjih si bomo ogledali, kako smo strukturirali najpomembnejše zaslonske maske.

5.4.1 Osnovni zaslon

Osnovni zaslon smo vertikalno razdelili na 3 dele. Zgornji del zaseda orodna vrstica, kjer so prikazane statusne ikone, trenuten čas in stanje baterije, podobno kot to najdemo na GUI v mobilnih napravah.

Srednji del je namenjen kazanju informacij o letu. Tu so zbrani podatki o trenutni hitrosti, višini, vertikalni hitrosti ter smeri in hitrosti vetra. Podatek o smeri vetra je podkrepljen še s simbolnim prikazom v zgornjem levem kotu zaslona. S pomočjo senc smo ustvarili občutek, da element ne leži na isti ravnini kot ozadje, ampak je od ozadja dvignjen. Največji del zaseda prikaz vertikalne hitrosti, ki je za pilota jadralnega letala eden najpomembnejših. Po priporočilih opisanih v poglavju 5.2 o dobri praksah v letalstvu, smo vertikalno hitrost prikazali v obliki digitalnega števca. Števec smo zaradi



Slika 5.5: Premiki med zaslonskimi maskami. Zaslonske maske na osnovnem nivoju so v odebeljenih okvirčkih.

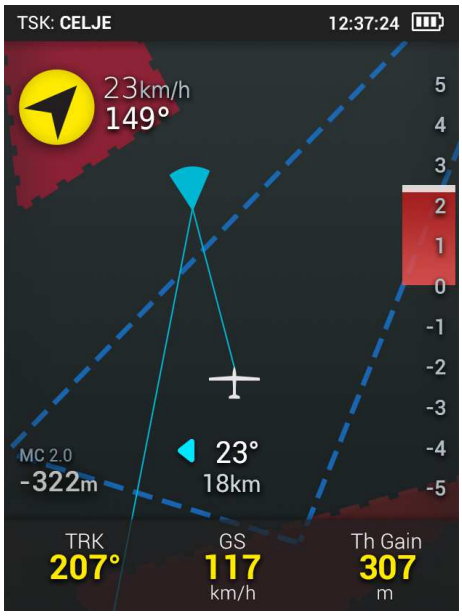
možnosti, ki nam jih dopušča izrisovanje na zaslon oblikovali tako, da je del števca raven, del pa zavit. Vrednosti so podkrepljene še z barvnim ozadjem, kjer rdeča simbolizira dviganje, modra pa spuščanje letala. Temnejša kot je barva, bolj hitro se letalo spuša oziroma dviga. Isti podatek smo tako uporabniku prikazali na 3 različne načine: numerično, z iglo in ozadjem. Poleg tega pa naprava uporabnika o trenutni vertikalni hitrosti opozarja še z zvočnim tonom. Najpomembnejši podatek prikazujemo tako, da vzbujamo dve različni čutili. Tudi pri prikazu vertikalne hitrosti smo s pomočjo senc ustvarili občutek tridimenzionalnega prostora na zaslonu.

Slika 5.6: Osnovni zaslon

Spodnji del zaslona prikazuje zgolj numerične podatke o letu. Vsak element v spodnji vrstici je sestavljen iz naslova, pri čemer uporabljamo kratice, ki so splošno znane v letalstvu [15], numeričnega prikaza vrednosti in enote v kateri je vrednost prikazana. S pritiskom na enega izmed podatkov, se lahko uporabnik odloči, kateri podatek o letu bi želel imeti prikazan na tem mestu. Izbira lahko med nekaj deset podatki, ki so na voljo. Spodnjo vrstico smo poimenovali NavBox.

5.4.2 Navigacijske strani

Navigacijska stran se pojavi na napravi v treh zaslonskih maskah. Od osnovnega zaslona smo prevzeli prikaz orodne in spodnje vrstice (NavBox). Spremenila se je torej le vsebina v srednjem delu zaslona. V orodni vrstici je levi del, kjer so bile sicer statusne ikone, nadomestil podatek o tem, na kateri navigacijski strani se uporabnik nahaja, ter ime točke kamor trenutno kaže navigacija. Strani, ki so na voljo so navigacija na letališče (APT), navigacija na točko (TP) in navigacija na nalogi (TSK).



Slika 5.7: Primer navigacijske strani na nalogi (TSK)

V srednjem delu zaslona pa prikazujemo pozicijo našega letala in stanje okolice. Pri tem vizualiziramo zračne prostore, ki so v območju in izrišemo navigacijske točke iz podatkovne baze. Zračni prostori so označeni s črtkanimi črtami. Območja, kjer ne smemo leteti so poleg tega obarvana rdeče. Si pa lahko uporabniki v nastavitvah nastavijo s kakšnimi barvami in polnili se izrisujejo zračni prostori na mapi, glede na vrsto zračnega prostora. V primeru navigacije na nalogi so na zaslonu označene tudi navigacij-

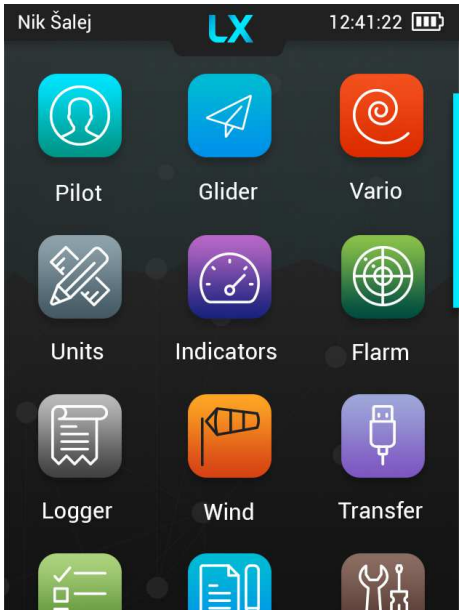
ske točke naloge skupaj s prostori okoli točk. Prostori okoli točk predstavljajo območje, ki ga mora pilot preleteti, da uspešno obleti izbrano točko. Prostor okoli točke je lahko krog ali kakršen koli izsek iz kroga. Točke so med sabo povezane s črtami. Na desni strani prikazujemo vertikalno hitrost letala. Da smo prihranili prostor smo se tukaj odločili, da vertikalno hitrost prikažemo s pomočjo traku. Uporabljamo pa enako barvno shemo kot na osnovnem zaslonu 5.6. Pod ikono letala prikazujemo oddaljenost in relativno smer do izbrane navigacijske točke. V levem spodnjem kotu pa je izpisana vrednost doleta. Vrednost doleta vedno prikažemo predznačeno s tem pa izboljšamo razumljivost numeričnega prikaza vrednosti. Kako izračunamo dolet smo si ogledali v poglavju 4.2.4.

5.4.3 Meni z nastavitvami

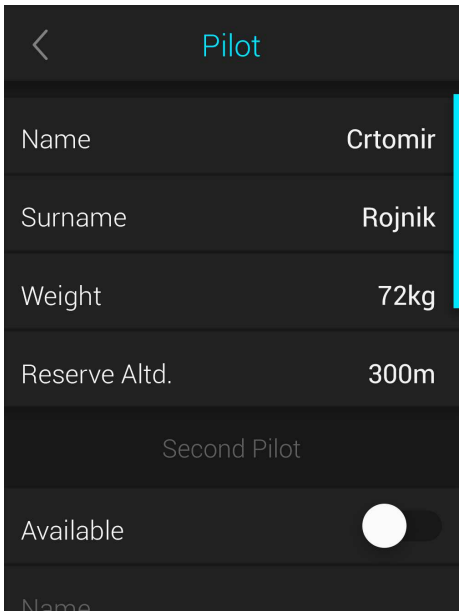
Zaslonske maske, kjer uporabnik dostopa do posamičnih nastavitev naprave, smo razdelili na dva dela. V prvem delu smo obravnavali zgolj izbiro področja nastavitev, v drugem pa dejansko spreminjanje nastavitev določenega področja. Pri gradnji GUI za nastavitve smo se v večini navezali na zaslonske maske, ki jih najdemo na modernih mobilnih napravah z zaslonom občutljivim na dotik.

Prvi nivo izbire področja smo zgradili kot mrežo podpisanih ikon. Z drsanjem po zaslonu uporabnik odkriva neprikazane ikone. Ob kliku na ikono se odpre druga stopnja nastavitev. Zaslonsko masko smo zgradili po predlogah Google Material Design [12]. Uporabili smo temno ozadje nanj pa smo položili vse ikone.

Drugi nivo nastavitev je prav tako, precej podoben zaslonskim maskam, ki jih najdemo na modernih mobilnih napravah. S tem smo novim uporabnikom olajšali spoznavanje z delovanjem naprave, saj so rokovanja s podobnimi GUI vajeni iz vsakdanjega življenja.



Slika 5.8: Osnovna zaslonska maska nastavitev.

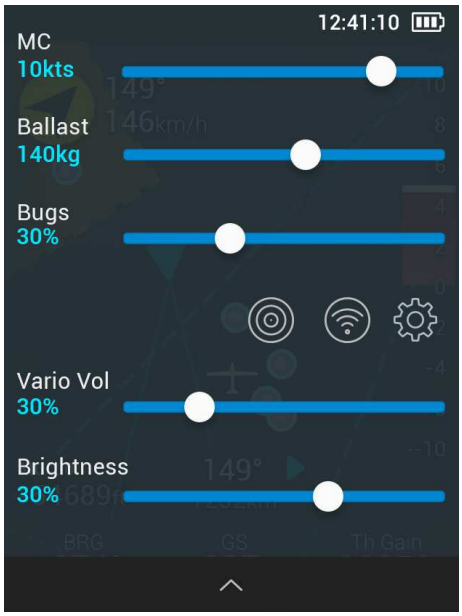


Slika 5.9: Primer zaslonske maske v nastavitvah.

5.4.4 Kretnje

Za prehajanje med glavnimi zaslonskimi maskami uporabljamo kretnjo, ko s prstom podrsamo iz ene strani zaslona na drugo (ang. swipe). Če želimo dostopati do zaslonske maske, ki je desno od trenutne potegnemo s prstom iz desne proti levi. V primeru, da želimo dostopati do zaslonske maske, ki je na levi od trenutne pa storimo ravno obratno.

V primeru, da zgornje kretnje uporabimo nad spodnjo vrstico na zaslonu, ki smo jo poimenovali NavBox, bomo prehajali med različnimi skupinami elementov, ki numerično prikazujejo podatke o letu.



Slika 5.10: Dostop do hitrih nastavitev s pomočjo kretnje.

Zgornja vrstica se pojavlja skoraj na vseh zaslonskih maskah, zato smo tudi nanjo vezali eno izmed kretenj. Če uporabnik povleče iz območja orodne vrstice proti dnu zaslona, bo odprl novo zaslonsko masko. Tukaj uporabnik hitro dostopa do najpomembnejših nastavitev. Ozadje v tem primeru ni polne barve, ampak je delno prosojno in s tem daje vtis, da gre za zaslonsko masko, ki se je odprla preko obstoječe vsebine. Tako uporabnik že vnaprej

ve, kaj lahko pričakuje na zaslonu, ko bo dialog izginil z zaslona. Primer zaslonske maske, ki se odpre ob uporabi opisane kretnje je viden na sliki 5.10.

Na navigacijskih straneh pa uporabljamo dvoprstno kretnjo. V primeru, da dva prsta približamo drug drugemu sprožimo akcijo, ki pomeni spremembo merila v katerem je prikazana karta. V primeru te kretnje bomo merilo povečali in na ta način na zaslonu prikazali večje območje okoli nas. Če pa uporabimo nasprotnjo kretnjo, kjer se z dvema prstoma oddaljimo drug od drugega, pa bomo merilo zmanjšali, s tem pa povzročili podrobnejši prikaz okolice okoli naše pozicije.

Sprehajanje po menujih, spreminjajnje vrednosti, uporaba tipkovnice, ki se izriše na zaslonu pa smo realizirali na povsem enak način, kot je to v GUI mobilnih naprav, ki smo jih preučili v poglavju 5.3.

Poglavje 6

Sklepne ugotovitve

V diplomski nalogi smo se ukvarjali z izdelavo jadralne navigacijske naprave. Razvoj takšne naprave se prične z zbiranjem funkcionalnih zahtev, ki jim mora naprava zadostiti. Pri tem smo imeli nekoliko lažje delo, saj se tudi sami ukvarjamo z jadralnim letenjem. Ko smo se odločili, kaj pričakujemo od naše naprave, se moramo odločiti kakšno strojno opremo bomo uporabili in kako jo bomo povezali v celoto. Izbira ustreznih elementov je ključna, saj nam ta določa in v veliki meri omejuje nadalnji razvoj programske opreme. Tudi v našem primeru je uporabljen procesor brez vgrajene enote MMU botroval k izbri operacijskega sistema, ki MMU ne potrebuje.

Še tako dobra kombinacija strojne in programske opreme pa pravzaprav ne pomeni nič, če informacij, ki jih naprava proizvede ne zmoremo učinkovito predstaviti uporabniku. V ta namen smo izvedli podrobno analizo grafičnih uporabniških vmesnikov uporabljenih v letalstvu in modernih mobilnih napravah, ki za vnos podatkov uporabljajo zaslon občutljiv na dotik. Z dobljenim znanjem smo zgradili uporabniški vmesnik.

Po ustreznem testiranju zgrajene naprave v zraku na reprezenativni množici pilotov, bi lahko dobljen uporabniški vmesnik še dodatno prilagodili. Predvidevamo, da bi lahko starejša populacija jadralnih pilotov imela na začetku nekaj težav z upravljanjem naprave. V ta namen bi lahko uvedli spremembe, kot je na primer izbira ustreznejše in predvsem večje tipografije.

Tudi izbiro barv bi lahko na določenih mestih bolj uskladili s praksami, ki so se uveljavile v navigacijskih napravah namenjenih motornim letalom.

Literatura

[1] Cristinel Ababei. Spi and sd cards. Lecture, Electrical Engineering Department, University at Buffalo, 2013.

[2] Apple. iOS Human Interface Guidelines. Dosegljivo: `https://developer.apple.com/ios/human-interface-guidelines/`. [Dostopano: 6. 7. 2016].

[3] Ali Bahrami. Flightcrew alerting. Advisory circular, U.S. Department of Transportation, Federal Aviation Administration, 2010.

[4] Bilinearna interpolacija. Dosegljivo: `https://en.wikipedia.org/wiki/Bilinear_interpolation`, 2016. [Dostopano: 5. 7. 2016].

[5] Janez Brezar...[et. Al.]. *JADRALNO letalstvo*. Letalska zveza Slovenije, Komisija za jadralno letenje, 1995.

[6] Capacitive sensing. Dosegljivo: `https://en.wikipedia.org/wiki/Capacitive_sensing`, 2016. [Dostopano: 5. 6. 2016].

[7] Rick Downs. Using resistive touch screen for human/machine interface. Technical report, Texas Instruments Incorporated, 2005.

[8] Jeffery E. Duven. Electronic flight displays. Advisory circular, U.S. Department of Transportation, Federal Aviation Administration, 2014.

[9] IGC FAI. Technical specification for gnss flight recorders. Technical report, International gliding commission, FAI, 2016.

[10] Bérard François. Measuring the Linear and Rotational User Precision in Touch Pointing. Proceedings of the 2012 ACM International Conference on Interactive Tabletops and Surfaces. New York., 2012.

[11] Satellite navigation. Dosegljivo: `https://en.wikipedia.org/wiki/Satellite_navigation`, 2016. [Dostopano: 5. 6. 2016].

[12] Google. Material design. Dosegljivo: `https://material.google.com`. [Dostopano: 23. 6. 2016].

[13] Dominik Gregl. *Nameravam postati PILOT jadralnega letala.* Samozaložba: Dominik Gregl, 2011.

[14] Graphical user interface. Dosegljivo: `https://en.wikipedia.org/wiki/Graphical_user_interface`, 2016. [Dostopano: 5. 6. 2016].

[15] ICAO. Doc 8400 procedures for air navigation services: Icao abbreviations and codes. Technical report, International Civil Aviation Organization, 2010.

[16] International Standard Atmosphere. Dosegljivo: `https://en.wikipedia.org/wiki/International_Standard_Atmosphere`, 2016. [Dostopano: 8. 7. 2016].

[17] SAE International. Sae arp5289a electronic aeronautical symbols (for depiction of navigation symbology). Technical report, Electrical Engineering Department, University at Buffalo, 2011.

[18] Vladimir Khusainov. Practical Advice on Running uClinux on Cortex-M3/M4. Dosegljivo: `http://electronicdesign.com/embedded/practical-advice-running-uclinux-cortex-m3m4`, 2012. [Dostopano: 2. 7. 2016].

[19] LPCZone. Get to know freertos from the creator! - designwest 2013. Dosegljivo: `https://youtu.be/1oagM_tEyeA`, 2013. [Dostopano: 1. 7. 2016].

[20] Treaty of Versailles. Dosegljivo: https://en.wikipedia.org/wiki/Treaty_of_Versailles, 2016. [Dostopano: 30. 4. 2016].

[21] Memory management unit. Dosegljivo: https://en.wikipedia.org/wiki/Memory_management_unit, 2016. [Dostopano: 1. 7. 2016].

[22] NASA. Shuttle Radar Topography Mission. Dosegljivo: http://www2.jpl.nasa.gov/srtm/, 2016. [Dostopano: 5. 5. 2016].

[23] GPS - NMEA sentence information. Dosegljivo: http://aprs.gids.nl/nmea. [Dostopano: 15. 6. 2016].

[24] NMEA data. Dosegljivo: http://www.gpsinformation.org/dale/nmea.htm. [Dostopano: 15. 6. 2016].

[25] Optical bonding. Dosegljivo: https://en.wikipedia.org/wiki/Optical_bonding, 2016. [Dostopano: 5. 6. 2016].

[26] Operating system. Dosegljivo: https://en.wikipedia.org/wiki/Operating_system, 2016. [Dostopano: 1. 7. 2016].

[27] Jennifer Preece and et al. *Interaction design*. John Wiley, Inc., 2002.

[28] Helmut Reichmann. *Cross-country soaring*. Thomson publications, 1978.

[29] Resistive vs Capacitive Touchscreen. Dosegljivo: https://techexplainer.wordpress.com/2012/04/02/resistive-vs-capacitive-touchscreen/, 2012. [Dostopano: 5. 6. 2016].

[30] Real-time operating system. Dosegljivo: https://en.wikipedia.org/wiki/Real-time_operating_system, 2016. [Dostopano: 1. 7. 2016].

[31] Adam Scot. The ultimate Android UI comparison. Dosegljivo: https://www.androidpit.com/android-ui-comparison, 2016. [Dostopano: 7. 7. 2016].

[32] Secure Digital. Dosegljivo: `https://en.wikipedia.org/wiki/Secure_Digital`, 2016. [Dostopano: 3. 5. 2016].

[33] Serial Peripheral Interface Bus. Dosegljivo: `https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus`, 2016. [Dostopano: 20. 5. 2016].

[34] Touchscreen. Dosegljivo: `https://en.wikipedia.org/wiki/Touchscreen`, 2016. [Dostopano: 5. 6. 2016].

[35] Edward R. Tufte. *The Visual Display of Quantitive Information.* Graphics Press LCC, 2007.

[36] Andrej Vrečer. Merjenje višine z uporabo digitalnega senzorja pritiska. Diplomska naloga, Fakulteta za elektrotehniko računalništvo in informatiko, Univerza v Mariboru, 2009.

[37] Jakob Zupančič. *Črtice o zrakoplovstvu in aviatiki.* Slovenska Šolska Matica, Ljubljana, 1911.